

Cesare Alippi
Marios Polycarpou
Christos Panayiotou
Georgios Ellinas (Eds.)

LNCS 5768

Artificial Neural Networks – ICANN 2009

19th International Conference
Limassol, Cyprus, September 2009
Proceedings, Part I

1
Part I

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Cesare Alippi
Marios Polycarpou
Christos Panayiotou
Georgios Ellinas (Eds.)

Artificial Neural Networks – ICANN 2009

19th International Conference
Limassol, Cyprus, September 14-17, 2009
Proceedings, Part I



Springer

Volume Editors

Cesare Alippi
Politecnico di Milano, Dipartimento di Elettronica
Piazza L. da Vinci 32, 20133 Milano, Italy
E-mail: alippi@elet.polimi.it

Marios Polycarpou
Christos Panayiotou
Georgios Ellinas
University of Cyprus, Department of Electrical and Computer Engineering
75 Kallipoleos Street 1678 Nicosia, Cyprus
E-mail: {mpolycar, christosp, gellinas}@ucy.ac.cy

Library of Congress Control Number: 2009933887

CR Subject Classification (1998): F.1, I.2, I.4, I.5, I.6, J.3, C.1.3, C.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-642-04273-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-04273-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2009
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12753574 06/3180 5 4 3 2 1 0

Preface

This volume is part of the two-volume proceedings of the 19th International Conference on Artificial Neural Networks (ICANN 2009), which was held in Cyprus during September 14–17, 2009. The ICANN conference is an annual meeting sponsored by the European Neural Network Society (ENNS), in cooperation with the International Neural Network Society (INNS) and the Japanese Neural Network Society (JNNS). ICANN 2009 was technically sponsored by the IEEE Computational Intelligence Society. This series of conferences has been held annually since 1991 in various European countries and covers the field of neurocomputing, learning systems and related areas.

Artificial neural networks provide an information-processing structure inspired by biological nervous systems. They consist of a large number of highly interconnected processing elements, with the capability of learning by example. The field of artificial neural networks has evolved significantly in the last two decades, with active participation from diverse fields, such as engineering, computer science, mathematics, artificial intelligence, system theory, biology, operations research, and neuroscience. Artificial neural networks have been widely applied for pattern recognition, control, optimization, image processing, classification, signal processing, etc.

In 2009, the ICANN conference was organized by the KIOS Research Center for Intelligent Systems and Networks and the Department of Electrical and Computer Engineering of the University of Cyprus. The conference was held at the seaside city of Limassol, which is the second largest city in Cyprus. The participants had the opportunity to enjoy the technical program, as well as the rich cultural heritage of Cyprus, whose 9,000-year cultural legacy has been at the crossroads of world history. Currently, Cyprus is a full member of the European Union that combines European culture with ancient enchantment.

Out of approximately 300 paper submissions to ICANN 2009, the Program Committee selected about 200 papers, which are published in the two volumes of these proceedings. The selection of the accepted papers was made after a thorough peer-review process, where each submission was evaluated by at least three reviewers. The submitted papers were authored by peer scholars coming from 47 countries, which geographically cover the whole planet (Europe, Middle East, Africa 69%; Asia/Pacific 18%; Americas 13%). The large number of accepted papers, variety of topics and high quality of submitted papers reflect the vitality of the field of artificial neural networks. In addition to the regular papers, the technical program featured keynote plenary lectures by worldwide renowned scholars, two tutorials on exciting new topics, two competitions on immunology and environmental toxicology prediction, and two workshops. One of the workshops was supported by the EU-sponsored COST Action “Intelligent Monitoring, Control and Security of Critical Infrastructure Systems” (IntelliCIS).

The two-volume proceedings contain papers on the following topics: Learning Algorithms; Computational Neuroscience; Hardware Implementations and Embedded Systems; Self Organization; Intelligent Control and Adaptive Systems; Neural and Hybrid

Architectures; Support Vector Machines; Recurrent Neural Networks; Neuro-informatics and Bioinformatics; Cognitive Machines; Data Analysis and Pattern Recognition; Signal and Time Series Processing; Applications; Neural Dynamics and Complex Systems; Vision and Image Processing; Neuro-evolution and Hybrid Techniques for Mobile Agents Control; Neural Control, Planning and Robotics Applications; Intelligent Tools and Methods for Multimedia Annotation; Critical Infrastructure Systems.

It is our pleasure to express our gratitude to everybody that contributed to the success of ICANN 2009. In particular, we thank the members of the Board of the European Neural Networks Society for entrusting us with the organization of the conference, as well as for their assistance during the preparation of ICANN 2009. Special thanks to the President of ENNS, Włodzisław Duch, who helped significantly toward the success of the conference. We would like to express our sincere gratitude to the members of the Program Committee and all the reviewers, who did a tremendous job under strict time limitations during the reviewing process. We thank the members of the Organizing Committee for the great effort in the organization of the conference and the members of the Local Organizing Committee for their assistance. We are grateful to the University of Cyprus (ECE Department) and the Cyprus Tourist Organization for their financial support. We thank the conference secretariat, Top Kinisis, and especially Christina Distrá, for their excellent and timely support in the organization of the conference. We are grateful to several researchers at the University of Cyprus and the Politecnico di Milano, who assisted in various ways in the organization of ICANN 2009, and especially Alexandros Kyriakides and Andreas Kartakoullis, who spent several days working on the formatting of the final proceedings, and Manuel Roveri, who addressed several software-related and procedural problems raised during the review process. We would also like to thank the publisher, Springer, for their co-operation in publishing the proceedings in the prestigious series of *Lecture Notes in Computer Science*. Finally, we thank all the authors who contributed to this volume for sharing their new ideas and results with the community. We hope that these ideas will generate further new ideas and innovations for the benefit of society and the environment.

July 2009

Cesare Alippi
Marios Polycarpou
Christos Panayiotou
Georgios Ellinas

Organization

Conference Chairs

General Co-chairs	Marios Polycarpou, University of Cyprus, Nicosia, Cyprus Christos Panayiotou, University of Cyprus, Nicosia, Cyprus
Program Chair	Cesare Alippi, Politecnico di Milano, Italy
Publications Chair	Georgios Ellinas, University of Cyprus, Nicosia, Cyprus
Finance Chair	Maria Michael, University of Cyprus, Nicosia, Cyprus
Local Arrangment Chair	Elias Kyriakides, University of Cyprus, Nicosia, Cyprus
Registration Chairs	Constantinos Pitris, University of Cyprus, Nicosia, Cyprus Manuel Roveri, Politecnico di Milano, Italy
Publicity Chairs	Derong Liu, University of Illinois at Chicago, USA Ouejdane Mejri, Politecnico di Milano, Italy

Program Committee

Angelo Alessandro	Angelo Università di Genova, Italy
Luís Alexandre	University of Beira Interior, Portugal
Bruno Apolloni	University of Milan, Italy
Amir Atiya	Cairo University, Egypt
Bartłomiej Beliczynski	Warsaw University of Technology, Poland
Monica Bianchini	Università degli Studi di Siena, Italy
Ivo Bukovsky	Czech Technical University in Prague, Czech Republic
Ke Chen	The University of Manchester, UK
Andrej Dobnikar	FRI, Slovenia
Gerard Dreyfus	ESPCI – Paristech, France
Wlodek Duch	Nicolaus Copernicus University, Poland
David A. Elizondo	De Montfort University, UK
Peter Erdi	Hungarian Academy of Sciences, Hungary
Pablo Estévez	University of Chile, Chile
Jay Farrell	University of California, Riverside, USA
Erol Gelenbe	Imperial College London, UK
Mark Girolami	University of Glasgow, UK
Barbara Hammer	Clausthal University of Technology, Germany
Tom Heskes	Radboud University Nijmegen, The Netherlands
Timo Honkela	Helsinki University of Technology, Finland
Amir Hussain	University of Stirling, UK
Giacomo Indiveri	University of Zurich - ETH Zurich, Switzerland

VIII Organization

Sylvain Jaume	Massachusetts Institute of Technology, USA
Yaochu Jin	Honda Research Institute Europe, Germany
Stefanos Kollias	National Technical University of Athens, Greece
Jan Koutník	IDSIA, Switzerland
Vera Kurkova	Academy of Sciences of the Czech Republic
Aristidis Likas	University of Ioannina, Greece
Derong Liu	University of Illinois at Chicago, USA
Francesco Marcelloni	University of Pisa, Italy
Thomas Martinetz	University of Lubeck, Germany
Matteo Matteucci	Politecnico di Milano, Italy
Ali Minai	University of Cincinnati, USA
Roman Neruda	Institute of Computer Science, Czech Republic
Erkki Oja	Helsinki University of Technology, Finland
Stanislaw Osowski	Warsaw University of Technology, Poland
Guenther Palm	University of Ulm, Germany
Thomas Parisini	University of Trieste, Italy
Constantinos Pattichis	University of Cyprus, Cyprus
Vincenzo Piuri	University of Milan, Italy
Mark Plumbley	Queen Mary University of London, UK
Manuel Roveri	Politecnico di Milano, Italy
George Rovithakis	Aristotle University of Thessaloniki, Greece
Leszek Rutkowski	Technical University of Czestochowa, Poland
Karl Sammut	Flinders University, Australia
Edgar Sanchez	Cinvestav, Mexico
Marcello Sanguineti	University of Genova, Italy
Fabio Scotti	University of Milan, Italy
Olli Simula	Helsinki University of Technology, Finland
Alessandro Sperduti	Università degli Studi di Padova, Italy
Andreas Stafylopatis	National Technical University of Athens, Greece
Johan Suykens	KULeuven, Belgium
John Taylor	King's College London, UK
Sergios Theodoridis	University of Athens, Greece
Marc Van Hulle	KU Leuven, Belgium
Marley Vellasco	Pontifícia Universidade Católica do Rio de Janeiro, Brazil
Michel Verleysen	Université Catholique de Louvain, Belgium
Alessandro E.P. Villa	Neuroheuristic, France
Jun Wang	The Chinese University of Hong Kong, Hong Kong
Zidong Wang	Brunel University, UK
Stefan Wermter	University of Sunderland, UK
Hujun Yin	The University of Manchester, UK
Michalis Zervakis	Technical University of Crete, Greece
Xiaodong Zhang	Wright State University, USA
Pedro Zufiria	Universidad Politécnica de Madrid, Spain
Rodolfo Zunino	University of Genoa, Italy

Local Organizing Committee

Constantinos Pattichis	University of Cyprus
Christos Schizas	University of Cyprus
Julius Georgiou	University of Cyprus
Vasilios Promponas	University of Cyprus
Elpida Keravnou	University of Cyprus
Petros Ioannou	Cyprus University of Technology
Chris Christodoulou	University of Cyprus
Andreas Pitsillides	University of Cyprus
Spyros Martzoukos	University of Cyprus
Theocharis Theocharides	University of Cyprus
Christoforos Hadjicostis	University of Cyprus
Nicolas Tsapatsoulis	Cyprus University of Technology

Referees

M. Adankon	A. Bahramisharif
L. Aijun	S. Bamford
F. Aiolfi	A. Bartczuk
T. Aksanova	S. Bassis
A. Al-Ani	G. Battistelli
A. Alanis	F. Bazso
A. Alessandri	C. Bechlioulis
L. Alexandre	B. Beliczynski
E. Alhoniemi	M. Bianchini
C. Alippi	M. Bicego
H. Allende	M. Biehl
C. Alzate	K. Blekas
F. Amigoni	S. Bohte
S. An	G. Boracchi
D. Andina	C. Borgelt
G. Andreoni	A. Borghese
M. Antonelli	A. Bors
B. Apolloni	P. Bouboulis
A. Araujo	Z. Buk
B. Arnonkijpanich	I. Bukovsky
Y. Asai	K. Bunte
S. Asteriadis	M. Bursa
M. Atencia	K. Cameron
A. Atiya	R. Camplani
Y. Avrithis	A. Cangelosi
D. Bacciu	A. Carvalho
A. Bagirov	M. Casey

I. Castelli	B. Denby
A. Castillo	M. Di Francesco
B. Castillo-Toledo	E. Di Iorio
D. Cavouras	V. Di Massa
M. Celebi	D. Dias
M. Cepek	M. Diesmann
C. Cervellera	A. Dobnikar
J. Chambers	S. Doboli
A. Chaves	E. Donchin
K. Chen	H. Dong
D. Chen	J. Drchal
L. Chen	G. Dreyfus
T. Chen	N. Drosopoulos
Z. Chengwen	P. D'Souza
M. Chetouani	P. Ducange
S. Choi	W. Duch
N. Chumerin	D. Durackova
R. Cierniak	P. Dziwnski
M. Cimino	M. Elgendi
J. Cires	D. Eliades
J.C. Claussen	D. Elizondo
M. Cococcioni	M. Elshaw
V. Colla	M. El-Telbany
S. Coniglio	P. Erdi
C. Constantinopoulos	M. Ernandes
S. Contassot-Vivier	H. Erwin
E. Corchado	P. Estevez
O. Cordon	A. Esuli
F. Corona	R. Ewerth
J.A. Costa	P. Földesi
A. Cruz	J. Farrell
E. Cuadros-Vargas	H. Fayed
E. Culurciello	J. Feng
M. Cuneo	G. Fenu
W. Czajewski	S. Ferrari
G. Da San Martino	K. Figueiredo
A. d'Avila Garcez	J. Fraile-Ardanuy
Z. Dai	D. François
S. Damas	L. Franco
T. Damoulas	G. Fumera
J. Dauwels	C. Fyfe
T. De Bie	M. Gabryel
A. De Paola	M. Gaggero
S. Decherchi	X-Z. Gao
Y. Demiris	E. Gelenbe

X. Georgiou	A. Keyhani
A. Ghio	F. Khan
G. Gigante	Y. Kim
G. Gini	T. Kimura
M. Girolami	A. Klamí
M. Giulioni	A. Knoblauch
G. Gnecco	O. Kohonen
E. Gomez-Ramirez	P. Koikkalainen
V. Gomez-Verdejo	S. Kollias
P. Gópych	M. Korytkowski
L. Graening	M. Koskela
M. Hagenbuchner	D. Kosmopoulos
B. Hammer	C. Kotropoulos
K. Harris	J. Koutník
F. He	K. Koutroumbas
X. He	O. Kovárik
M. Hernanez	C. Krier
A. Herrero	D. Krusinski
T. Heskes	F. Kurfess
L. Hey	V. Kurkova
H. Hidalgo	G. La Camera
M. Hilairet	K. Lagus
T. Honkela	J. Lampinen
L. Hontoria	C. Laoudias
P. Hoyer	A. Lazaric
E. Hruschka	L. Y. Lean Yu
J. Hu	J. Lee
T. Huang	A. Lendasse
G.-B. Huang	L. Li
H. Huang	W. Li
C. Hung	A. Likas
A. Hussain	B. Linares-Barranco
A. Ilin	T. Lindh-Knuutila
G. Indiveri	M. Lippi
B. Indurkhya	D. Liu
S. Jaume	J. Liu
J. Jerez	W. Liu
Y. Jin	Y. Liu
M. Jirina	D. Loiacono
M. Jirina	P.-Y. Louis
A. Kaban	G. Loukas
M. Kaipainen	J. Lücke
J. Karhunen	L. Ma
K. Karpouzis	D. Macciò
N. Kasabov	M. MacIver

M. Mahfouf	M. Ortolani
M. Manry	S. Osowski
N. Manyakov	Y. Oussar
A. Marakakis	N. Oza
F. Marcelloni	T. Pahikkala
R. Marek	A. Paiva
M. Markou	G. Palm
U. Markowska-Kaczmar	P. Panagi
T. Martinetz	D. Panagiotopoulos
T. Martinez-Marin	C. Panayiotou
A. Marullo	T. Papini
P. Masci	T. Parisini
S. Maskell	K. Parsopoulos
P. Massobrio	E. Pasero
F. Masulli	C. Pattichis
G. Matsopoulos	G. Pazienza
M. Matteucci	F. Pellegrino
E. Merenyi	S. Petridis
M. Michaelides	A. Petrosino
A. Micheli	B. Phillips
A. Minai	F. Piazza
S. Mitra	V. Piuri
N. Mitrakis	M. Plumbley
U. Modigliani	T. Poggi
F. Morabito	M. Polycarpou
G. Morgavi	F. Prieto
S. Moustakidis	A. Pucci
J. Murray	K. Raftopoulos
P. Mylonas	K. Raivio
S. Nara	F. Ramos-Corchado
M. Nelson	M.A. Ranzato
A. Neme	K. Rapantzikos
R. Neruda	M. Restelli
M. Niazi	A. Revonsuo
K. Nikita	R. Riaza
C. Nikou	B. Ribeiro
D. Novak	L. Ricalde
R. Nowicki	S. Ridella
K. Ntalianis	L. Rigutini
D. Oberhoff	F. Rivas
E. Oja	T. Rodemann
K. Okada	R. Rodriguez
K. Okuma	S. Rogers
C. Orovas	J. Ronda
J. Ortiz de Lazcano Lobato	F. Rossi

M. Roveri
S. Rovetta
G. Rovithakis
G. Ruthenbeck
L. Rutkowski
J. Ruz
D. Saez
K. Saito
M. Salomon
K. Sammut
E. Sanchez
M. Sanguineti
J. Santos
L. Sarti
S. Sayeed
G. Schaefer
R. Scherer
F.-M. Schleif
P. Schneider
F. Schwenker
F. Scotti
G. Sen Gupta
R. Setiono
E. Severin
I.-U.-H. Shah
I. Shah
V. Shaposhnyk
B. Shen
Y. Shen
H. Shimazu
N. Siebel
L. Silva
O. Simula
G. Siolas
M. Skrbek
K. Slavakis
S. Slusny
L. Smetana
Z. Somogyvari
Q. Song
A. Sorjamaa
U. Spagnolini
A. Sperduti
E. Spyrou
S. Squartini
A. Stafylopatis
G. Stamou
J. Starczewski
J. Stastny
V. Stathopoulos
D. Stavrakoudis
N. Steele
M. Strickert
J. Suykens
K. Swingler
F. Talantzis
Y. Tang
R. Tanscheit
H. Tarik
J. Taylor
A. Tefas
S. Theodoridis
T. Theoharis
C. Thiel
S. Timotheou
P. Tino
M. Titsias
J. Togelius
E. Trentin
N. Tsapatsoulis
S. Tzikopoulos
J. Vayrynen
M. Valle
H. Valpola
M. Van Hulle
M. Vellasco
M. Verleysen
J. Vesanto
A. Villa
T. Villmann
G. von der Emde
B. Vrusias
M. Wallace
J. Wang
Z. Wang
X. Wang
Y. Wang
Y.X. Wang
Z.-W. Wang
C. Weber

XIV Organization

Q. Wei	S. Zafeiriou
G. Wei	P. Zegers
H. Wei	S. Zeng
T. Wennekters	M. Zervakis
S. Wermter	X. Zhang
D. Wierstra	Q. Zhao
T. Windeatt	Z. Zhao
T. Xiang	M. Zhong
T. Yagi	S. Zhou
Z. Yang	J. Zhou
C. Yang	Z.-H. Zhou
Y. Yang	P. Zufiria
H. Yin	R. Zunino

Table of Contents – Part I

Learning Algorithms

Mutual Information Based Initialization of Forward-Backward Search for Feature Selection in Regression Problems	1
<i>Alberto Guillén, Antti Sorjamaa, Gines Rubio, Amaury Lendasse, and Ignacio Rojas</i>	
Kernel Learning for Local Learning Based Clustering	10
<i>Hong Zeng and Yiu-ming Cheung</i>	
Projective Nonnegative Matrix Factorization with α -Divergence	20
<i>Zhirong Yang and Erkki Oja</i>	
Active Generation of Training Examples in Meta-Regression	30
<i>Ricardo B.C. Prudêncio and Teresa B. Ludermir</i>	
A Maximum-Likelihood Connectionist Model for Unsupervised Learning over Graphical Domains	40
<i>Edmondo Trentin and Leonardo Rigutini</i>	
Local Feature Selection for the Relevance Vector Machine Using Adaptive Kernel Learning	50
<i>Dimitris Tzikas, Aristidis Likas, and Nikolaos Galatsanos</i>	
MINLIP: Efficient Learning of Transformation Models	60
<i>Vanya Van Belle, Kristiaan Pelckmans, Johan A.K. Suykens, and Sabine Van Huffel</i>	
Efficient Uncertainty Propagation for Reinforcement Learning with Limited Data	70
<i>Alexander Hans and Steffen Udluft</i>	
Optimal Training Sequences for Locally Recurrent Neural Networks	80
<i>Krzysztof Patan and Maciej Patan</i>	
Statistical Instance-Based Ensemble Pruning for Multi-class Problems	90
<i>Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez</i>	
Robustness of Kernel Based Regression: A Comparison of Iterative Weighting Schemes	100
<i>Kris De Brabanter, Kristiaan Pelckmans, Jos De Brabanter, Michiel Debruyne, Johan A.K. Suykens, Mia Hubert, and Bart De Moor</i>	

Mixing Different Search Biases in Evolutionary Learning Algorithms	111
<i>Kristina Davoian and Wolfram-M. Lippe</i>	
Semi-supervised Learning for Regression with Co-training by Committee	121
<i>Mohamed Farouk Abdel Hady, Friedhelm Schwenker, and Günther Palm</i>	
An Analysis of Meta-learning Techniques for Ranking Clustering Algorithms Applied to Artificial Data	131
<i>Rodrigo G.F. Soares, Teresa B. Ludermir, and Francisco A.T. De Carvalho</i>	
Probability-Based Distance Function for Distance-Based Classifiers	141
<i>Cezary Dendek and Jacek Mańdziuk</i>	
Constrained Learning Vector Quantization or Relaxed k-Separability	151
<i>Marek Grochowski and Włodzisław Duch</i>	
Minimization of Quadratic Binary Functional with Additive Connection Matrix	161
<i>Leonid Litinskii</i>	
Mutual Learning with Many Linear Perceptrons: On-Line Learning Theory	171
<i>Kazuyuki Hara, Yoichi Nakayama, Seiji Miyoshi, and Masato Okada</i>	
Computational Neuroscience	
Synchrony State Generation in Artificial Neural Networks with Stochastic Synapses	181
<i>Karim El-Laithy and Martin Bogdan</i>	
Coexistence of Cell Assemblies and STDP	191
<i>Florian Hauser, David Bouchain, and Günther Palm</i>	
Controlled and Automatic Processing in Animals and Machines with Application to Autonomous Vehicle Control	198
<i>Kevin Gurney, Amir Hussain, Jon Chambers, and Rudwan Abdullah</i>	
Multiple Sound Source Localisation in Reverberant Environments Inspired by the Auditory Midbrain	208
<i>Jindong Liu, David Perez-Gonzalez, Adrian Rees, Harry Erwin, and Stefan Wermter</i>	
A Model of Neuronal Specialization Using Hebbian Policy-Gradient with “Slow” Noise	218
<i>Emmanuel Daucé</i>	

How Bursts Shape the STDP Curve in the Presence/Absence of GABAergic Inhibition	229
<i>Vassilis Cutsuridis, Stuart Cobb, and Bruce P. Graham</i>	
Optimizing Generic Neural Microcircuits through Reward Modulated STDP	239
<i>Prashant Joshi and Jochen Triesch</i>	
Calcium Responses Model in Striatum Dependent on Timed Input Sources	249
<i>Takashi Nakano, Junichiro Yoshimoto, Jeff Wickens, and Kenji Doya</i>	
Independent Component Analysis Aided Diagnosis of Cuban Spino Cerebellar Ataxia 2	259
<i>Rodolfo V. García, Fernando Rojas, Jesús González, Belén San Román, Olga Valenzuela, Alberto Prieto, Luis Velázquez, and Roberto Rodríguez</i>	
Hippocampus, Amygdala and Basal Ganglia Based Navigation Control	267
<i>Ansgar Koene and Tony J. Prescott</i>	
A Framework for Simulation and Analysis of Dynamically Organized Distributed Neural Networks	277
<i>Vladyslav Shaposhnyk, Pierre Dutoit, Victor Contreras-Lámus, Stephen Perrig, and Alessandro E.P. Villa</i>	
Continuous Attractors of Lotka-Volterra Recurrent Neural Networks....	287
<i>Haixian Zhang, Jiali Yu, and Zhang Yi</i>	
Learning Complex Population-Coded Sequences.....	296
<i>Kiran V. Byadarhaly, Mithun Perdoor, Suresh Vasa, Emmanuel Fernandez, and Ali A. Minai</i>	
Structural Analysis on STDP Neural Networks Using Complex Network Theory	306
<i>Hideyuki Kato, Tohru Ikeguchi, and Kazuyuki Aihara</i>	
Time Coding of Input Strength Is Intrinsic to Synapses with Short Term Plasticity	315
<i>Márton A. Hajnal</i>	
Information Processing and Timing Mechanisms in Vision.....	325
<i>Andrea Guazzini, Pietro Lió, Andrea Passarella, and Marco Conti</i>	
Review of Neuron Types in the Retina: Information Models for Neuroengineering	335
<i>German D. Valderrama-Gonzalez, T.M. McGinnity, Liam Maguire, and QingXiang Wu</i>	

Brain Electric Microstate and Perception of Simultaneously Audiovisual Presentation	345
<i>Wichian Sittiprapaporn and Jun Soo Kwon</i>	
A Model for Neuronal Signal Representation by Stimulus-Dependent Receptive Fields	356
<i>José R.A. Torreão, João L. Fernandes, and Silvia M.C. Victer</i>	
Hardware Implementations and Embedded Systems	
Area Chip Consumption by a Novel Digital CNN Architecture for Pattern Recognition	363
<i>Emil Raschman and Daniela Ďuračková</i>	
Multifold Acceleration of Neural Network Computations Using GPU	373
<i>Alexander Guzhva, Sergey Dolenko, and Igor Persiantsev</i>	
Training Recurrent Neural Network Using Multistream Extended Kalman Filter on Multicore Processor and Cuda Enabled Graphic Processor Unit	381
<i>Michal Černanský</i>	
A Non-subtraction Configuration of Self-similitude Architecture for Multiple-Resolution Edge-Filtering CMOS Image Sensor	391
<i>Norihiro Takahashi and Tadashi Shibata</i>	
Current-Mode Computation with Noise in a Scalable and Programmable Probabilistic Neural VLSI System	401
<i>Chih-Cheng Lu and H. Chen</i>	
Minimising Contrastive Divergence with Dynamic Current Mirrors	410
<i>Chih-Cheng Lu and H. Chen</i>	
Spiking Neural Network Self-configuration for Temporal Pattern Recognition Analysis	421
<i>Josep L. Rosselló, Ivan de Paúl, Vincent Canals, and Antoni Morro</i>	
Image Recognition in Analog VLSI with On-Chip Learning	429
<i>Gonzalo Carvajal, Waldo Valenzuela, and Miguel Figueroa</i>	
Behavior Modeling by Neural Networks	439
<i>Lambert Spaanenburg, Mona Akbarniai Tehrani, Richard Kleihorst, and Peter B.L. Meijer</i>	
Statistical Parameter Identification of Analog Integrated Circuit Reverse Models	449
<i>Bruno Apolloni, Simone Bassis, Cristian Mesiano, Salvatore Rinaudo, Angelo Ciccazzo, and Angelo Marotta</i>	

A New FGMOST Euclidean Distance Computational Circuit Based on Algebraic Mean of the Input Potentials	459
<i>Cosmin Radu Popa</i>	
FPGA Implementation of Support Vector Machines for 3D Object Identification	467
<i>Marta Ruiz-Llata and Mar Yébenes-Calvino</i>	
Reconfigurable MAC-Based Architecture for Parallel Hardware Implementation on FPGAs of Artificial Neural Networks Using Fractional Fixed Point Representation	475
<i>Rodrigo Martins da Silva, Nadia Nedjah, and Luiza de Macedo Mourelle</i>	

Self Organization

A Two Stage Clustering Method Combining Self-Organizing Maps and Ant K-Means	485
<i>Jefferson R. Souza, Teresa B. Ludermir, and Leandro M. Almeida</i>	
Image Theft Detection with Self-Organising Maps	495
<i>Philip Prentis, Mats Sjöberg, Markus Koskela, and Jorma Laaksonen</i>	
Improved Kohonen Feature Map Associative Memory with Area Representation for Sequential Analog Patterns	505
<i>Tomonori Shirotori and Yuko Osana</i>	
Surface Reconstruction Method Based on a Growing Self-Organizing Map	515
<i>Renata L.M.E. do Rego, Hansenclever F. Bassani, Daniel Filgueiras, and Aluizio F.R. Araujo</i>	
Micro-SOM: A Linear-Time Multivariate Microaggregation Algorithm Based on Self-Organizing Maps	525
<i>Agusti Solanas, Arnau Gavalda, and Robert Rallo</i>	
Identifying Clusters Using Growing Neural Gas: First Results	536
<i>Riccardo Rizzo and Alfonso Urso</i>	
Hierarchical Architecture with Modular Network SOM and Modular Reinforcement Learning	546
<i>Masumi Ishikawa and Kosuke Ueno</i>	
Hybrid Systems for River Flood Forecasting Using MLP, SOM and Fuzzy Systems	557
<i>Ivna Valen��a and Teresa Ludermir</i>	

Topographic Mapping of Astronomical Light Curves via a Physically Inspired Probabilistic Model	567
<i>Nikolaos Gianniotis, Peter Tiňo, Steve Spreckley, and Somak Raychaudhury</i>	
Generalized Self-Organizing Mixture Autoregressive Model for Modeling Financial Time Series	577
<i>Hujun Yin and He Ni</i>	
Self-Organizing Map Simulations Confirm Similarity of Spatial Correlation Structure in Natural Images and Cortical Representations	587
<i>A. Ravishankar Rao and Guillermo Cecchi</i>	
Intelligent Control and Adaptive Systems	
Height Defuzzification Method on L^∞ Space	598
<i>Takashi Mitsuishi and Yasunari Shidama</i>	
An Additive Reinforcement Learning	608
<i>Takeshi Mori and Shin Ishii</i>	
Neural Spike Suppression by Adaptive Control of an Unknown Steady State	618
<i>Arūnas Tamaševičius, Elena Tamaševičiūtė, Gytis Mykolaitis, Skaidra Bumelienė, Raimundas Kirvaitis, and Ruedi Stoop</i>	
Combined Mechanisms of Internal Model Control and Impedance Control under Force Fields	628
<i>Naoki Tomi, Manabu Gouko, and Koji Ito</i>	
Neural Network Control of Unknown Nonlinear Systems with Efficient Transient Performance	638
<i>Elias B. Kosmatopoulos, Diamantis Manolis, and M. Papageorgiou</i>	
High-Order Fuzzy Switching Neural Networks: Application to the Tracking Control of a Class of Uncertain SISO Nonlinear Systems	648
<i>Haris E. Psillakis</i>	
Neural and Hybrid Architectures	
A Guide for the Upper Bound on the Number of Continuous-Valued Hidden Nodes of a Feed-Forward Network	658
<i>Rua-Huan Tsaih and Yat-wah Wan</i>	
Comparative Study of the CG and HBF ODEs Used in the Global Minimization of Nonconvex Functions	668
<i>Amit Bhaya, Fernando A. Pazos, and Eugenius Kaszkurewicz</i>	

On the Knowledge Organization in Concept Formation: An Exploratory Cognitive Modeling Study.....	678
<i>Toshihiko Matsuka, Hidehito Honda, Arieta Chouchourelou, and Sachiko Kiyokawa</i>	
Dynamics of Incremental Learning by VSF-Network	688
<i>Yoshitsugu Kakemoto and Shinchi Nakasuka</i>	
Kernel CMAC with Reduced Memory Complexity	698
<i>Gábor Horváth and Kristóf Gáti</i>	
Model Complexity of Neural Networks and Integral Transforms	708
<i>Věra Kůrková</i>	
Function Decomposition Network	718
<i>Yevgeniy Bodyanskiy, Sergiy Popov, and Mykola Titov</i>	
Improved Storage Capacity in Correlation Matrix Memories Storing Fixed Weight Codes	728
<i>Stephen Hobson and Jim Austin</i>	
Multiagent Reinforcement Learning with Spiking and Non-Spiking Agents in the Iterated Prisoner’s Dilemma	737
<i>Vassilis Vassiliades, Aristodemos Cleanthous, and Chris Christodoulou</i>	
Unsupervised Learning in Reservoir Computing: Modeling Hippocampal Place Cells for Small Mobile Robots	747
<i>Eric A. Antonelo and Benjamin Schrauwen</i>	
Switching Hidden Markov Models for Learning of Motion Patterns in Videos	757
<i>Matthias Höffken, Daniel Oberhoff, and Marina Kolesnik</i>	
Multimodal Belief Integration by HMM/SVM-Embedded Bayesian Network: Applications to Ambulating PC Operation by Body Motions and Brain Signals	767
<i>Yasuo Matsuyama, Fumiya Matsushima, Youichi Nishida, Takashi Hatakeyama, Nimiko Ochiai, and Shogo Aida</i>	
A Neural Network Model of Metaphor Generation with Dynamic Interaction	779
<i>Asuka Terai and Masanori Nakagawa</i>	
Almost Random Projection Machine	789
<i>Włodzisław Duch and Tomasz Maszczyk</i>	
Optimized Learning Vector Quantization Classifier with an Adaptive Euclidean Distance	799
<i>Renata M.C.R. de Souza and Telmo de M. Silva Filho</i>	

Efficient Parametric Adjustment of Fuzzy Inference System Using Error Backpropagation Method	807
<i>Ivan da Silva and Rogerio Flauzino</i>	
Neuro-fuzzy Rough Classifier Ensemble	817
<i>Marcin Korytkowski, Robert Nowicki, and Rafał Scherer</i>	
Combining Feature Selection and Local Modelling in the KDD Cup 99 Dataset	824
<i>Iago Porto-Díaz, David Martínez-Rego, Amparo Alonso-Betanzos, and Oscar Fontenla-Romero</i>	
An Automatic Parameter Adjustment Method of Pulse Coupled Neural Network for Image Segmentation	834
<i>Masato Yonekawa and Hiroaki Kurokawa</i>	
Pattern Identification by Committee of Potts Perceptrons	844
<i>Vladimir Kryzhanovsky</i>	

Support Vector Machine

Is Primal Better Than Dual	854
<i>Shigeo Abe</i>	
A Fast BMU Search for Support Vector Machine	864
<i>Wataru Kasai, Yutaro Tobe, and Osamu Hasegawa</i>	
European Option Pricing by Using the Support Vector Regression Approach	874
<i>Panayiotis C. Andreou, Chris Charalambous, and Spiros H. Martzoukos</i>	
Learning SVMs from Sloppily Labeled Data	884
<i>Guillaume Stempfel and Liva Ralaivola</i>	
The GMM-SVM Supervector Approach for the Recognition of the Emotional Status from Speech	894
<i>Friedhelm Schwenker, Stefan Scherer, Yasmine M. Magdi, and Günther Palm</i>	
A Simple Proof of the Convergence of the SMO Algorithm for Linearly Separable Problems	904
<i>Jorge López and José R. Dorronsoro</i>	
Spanning SVM Tree for Personalized Transductive Learning	913
<i>Shaoning Pang, Tao Ban, Youki Kadobayashi, and Nik Kasabov</i>	
Improving Text Classification Performance with Incremental Background Knowledge	923
<i>Catarina Silva and Bernardete Ribeiro</i>	

Empirical Study of the Universum SVM Learning for High-Dimensional Data	932
<i>Vladimir Cherkassky and Wuyang Dai</i>	

Relevance Feedback for Content-Based Image Retrieval Using Support Vector Machines and Feature Selection	942
<i>Apostolos Marakakis, Nikolaos Galatsanos, Aristidis Likas, and Andreas Stafylopatis</i>	

Recurrent Neural Network

Understanding the Principles of Recursive Neural Networks: A Generative Approach to Tackle Model Complexity	952
<i>Alejandro Chinea</i>	

An EM Based Training Algorithm for Recurrent Neural Networks	964
<i>Jan Unkelbach, Sun Yi, and Jürgen Schmidhuber</i>	

Modeling D_{st} with Recurrent EM Neural Networks	975
<i>Derrick Takeshi Mirikitani and Lahcen Ouarbya</i>	

On the Quantification of Dynamics in Reservoir Computing	985
<i>David Verstraeten and Benjamin Schrauwen</i>	

Solving the CLM Problem by Discrete-Time Linear Threshold Recurrent Neural Networks	995
<i>Lei Zhang, Pheng Ann Heng, and Zhang Yi</i>	

Scalable Neural Networks for Board Games	1005
<i>Tom Schaul and Jürgen Schmidhuber</i>	

Reservoir Size, Spectral Radius and Connectivity in Static Classification Problems	1015
<i>Luís A. Alexandre and Mark J. Embrechts</i>	

Author Index	1025
---------------------------	------

Table of Contents – Part II

Neuroinformatics and Bioinformatics

Epileptic Seizure Prediction and the Dimensionality Reduction Problem	1
<i>André Ventura, João M. Franco, João P. Ramos, Bruno Direito, and António Dourado</i>	
Discovering Diagnostic Gene Targets and Early Diagnosis of Acute GVHD Using Methods of Computational Intelligence over Gene Expression Data	10
<i>Maurizio Fiasché, Anju Verma, Maria Cuzzola, Pasquale Iacopino, Nikola Kasabov, and Francesco C. Morabito</i>	
Mining Rules for the Automatic Selection Process of Clustering Methods Applied to Cancer Gene Expression Data	20
<i>André C.A. Nascimento, Ricardo B.C. Prudêncio, Marcílio C.P. de Souto, and Ivan G. Costa</i>	
A Computational Retina Model and Its Self-adjustment Property	30
<i>Hui Wei and XuDong Guan</i>	

Cognitive Machines

Mental Simulation, Attention and Creativity	40
<i>Matthew Hartley and John G. Taylor</i>	
BSDT Atom of Consciousness Model, AOCM: The Unity and Modularity of Consciousness	54
<i>Petro Gopych</i>	
Generalized Simulated Annealing and Memory Functioning in Psychopathology	65
<i>Roseli S. Wedemann, Luís Alfredo V. de Carvalho, and Raul Donangelo</i>	
Algorithms for Structural and Dynamical Polychronous Groups Detection	75
<i>Régis Martinez and Hélène Paugam-Moisy</i>	
Logics and Networks for Human Reasoning	85
<i>Steffen Hölldobler and Carrolinne Dewi Puspa Kencana Ramli</i>	

Data Analysis and Pattern Recognition

Simbed: Similarity-Based Embedding	95
<i>John A. Lee and Michel Verleysen</i>	
PCA-Based Representations of Graphs for Prediction in QSAR Studies	105
<i>Riccardo Cardin, Lisa Michielan, Stefano Moro, and Alessandro Sperduti</i>	
Feature Extraction Using Linear and Non-linear Subspace Techniques	115
<i>Ana R. Teixeira, Ana Maria Tomé, and E.W. Lang</i>	
Classification Based on Combination of Kernel Density Estimators	125
<i>Mateusz Kobos and Jacek Marídziuk</i>	
Joint Approximate Diagonalization Utilizing AIC-Based Decision in the Jacobi Method	135
<i>Yoshitatsu Matsuda and Kazunori Yamaguchi</i>	
Newtonian Spectral Clustering	145
<i>Konstantinos Blekas, K. Christodoulidou, and I.E. Lagaris</i>	
Bidirectional Clustering of MLP Weights for Finding Nominally Conditioned Polynomials	155
<i>Yusuke Tanahashi and Ryohei Nakano</i>	
Recognition of Properties by Probabilistic Neural Networks	165
<i>Jiří Grim and Jan Hora</i>	
On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification	175
<i>Jorge M. Santos and Mark Embrechts</i>	
Profiling of Mass Spectrometry Data for Ovarian Cancer Detection Using Negative Correlation Learning	185
<i>Shan He, Huanhuan Chen, Xiaoli Li, and Xin Yao</i>	
Kernel Alignment k -NN for Human Cancer Classification Using the Gene Expression Profiles	195
<i>Manuel Martín-Merino and Javier de las Rivas</i>	
Convex Mixture Models for Multi-view Clustering	205
<i>Grigorios Tzortzis and Aristidis Likas</i>	
Strengthening the Forward Variable Selection Stopping Criterion	215
<i>Luis Javier Herrera, G. Rubio, H. Pomares, B. Paechter, A. Guillén, and I. Rojas</i>	

Features and Metric from a Classifier Improve Visualizations with Dimension Reduction	225
<i>Elina Parviainen and Aki Vehtari</i>	
Fuzzy Cluster Validation Using the Partition Negentropy Criterion	235
<i>Luis F. Lago-Fernández, Manuel Sánchez-Montaños, and Fernando Corbacho</i>	
Bayesian Estimation of Kernel Bandwidth for Nonparametric Modelling	245
<i>Adrian G. Bors and Nikolaos Nasios</i>	
Using Kernel Basis with Relevance Vector Machine for Feature Selection	255
<i>Frédéric Suard and David Mercier</i>	
Acquiring and Classifying Signals from Nanopores and Ion-Channels ...	265
<i>Bharatan Konnanath, Prasanna Sattigeri, Trupthi Mathew, Andreas Spanias, Shalini Prasad, Michael Goryll, Trevor Thornton, and Peter Knee</i>	
Hand-Drawn Shape Recognition Using the SVM'ed Kernel	275
<i>Khaled S. Refaat and Amir F. Atiya</i>	
Selective Attention Improves Learning	285
<i>Antti Yli-Krekola, Jaakko Särelä, and Harri Valpola</i>	
Signal and Time Series Processing	
Multi-stage Algorithm Based on Neural Network Committee for Prediction and Search for Precursors in Multi-dimensional Time Series	295
<i>Sergey Dolenko, Alexander Guzhva, Igor Persiantsev, and Julia Shugai</i>	
Adaptive Ensemble Models of Extreme Learning Machines for Time Series Prediction	305
<i>Mark van Heeswijk, Yoan Miche, Tiina Lindh-Knuutila, Peter A.J. Hilbers, Timo Honkela, Erkki Oja, and Amaury Lendasse</i>	
Identifying Customer Profiles in Power Load Time Series Using Spectral Clustering	315
<i>Carlos Alzate, Marcelo Espinoza, Bart De Moor, and Johan A.K. Suykens</i>	
Transformation from Complex Networks to Time Series Using Classical Multidimensional Scaling	325
<i>Yuta Haraguchi, Yutaka Shimada, Tohru Ikeguchi, and Kazuyuki Aihara</i>	

Predicting the Occupancy of the HF Amateur Service with Neural Network Ensembles	335
<i>Harris Papadopoulos and Haris Haralambous</i>	
An Associated-Memory-Based Stock Price Predictor	345
<i>Shigeki Nagaya, Zhang Chenli, and Osamu Hasegawa</i>	
A Case Study of ICA with Multi-scale PCA of Simulated Traffic Data	358
<i>Shengkun Xie, Pietro Lió, and Anna T. Lawniczak</i>	
Decomposition Methods for Detailed Analysis of Content in ERP Recordings	368
<i>Vasiliki Iordanidou, Kostas Michalopoulos, Vangelis Sakkalis, and Michalis Zervakis</i>	
Outlier Analysis in BP/RP Spectral Bands	378
<i>Diego Ordóñez, Carlos Dafonte, Minia Manteiga, and Bernardino Arcay</i>	
ANNs and Other Machine Learning Techniques in Modelling Models' Uncertainty	387
<i>Durga Lal Shrestha, Nagendra Kayastha, and Dimitri P. Solomatine</i>	
Comparison of Adaptive Algorithms for Significant Feature Selection in Neural Network Based Solution of the Inverse Problem of Electrical Prospecting	397
<i>Sergey Dolenko, Alexander Guzhva, Eugeny Obornev, Igor Persiantsev, and Mikhail Shimelevich</i>	
Efficient Optimization of the Parameters of LS-SVM for Regression versus Cross-Validation Error	406
<i>Ginés Rubio, Héctor Pomares, Ignacio Rojas, Luis Javier Herrera, and Alberto Guillén</i>	
Applications	
Noiseless Independent Factor Analysis with Mixing Constraints in a Semi-supervised Framework. Application to Railway Device Fault Diagnosis	416
<i>Etienne Côme, Latifa Oukhellou, Thierry Denœux, and Patrice Aknin</i>	
Speech Hashing Algorithm Based on Short-Time Stability	426
<i>Ning Chen and Wang-Gen Wan</i>	
A New Method for Complexity Reduction of Neuro-fuzzy Systems with Application to Differential Stroke Diagnosis	435
<i>Krzysztof Cpałka, Olga Rebrova, and Leszek Rutkowski</i>	

LS Footwear Database - Evaluating Automated Footwear Pattern Analysis	445
<i>Maria Pavlou and Nigel M. Allinson</i>	
Advanced Integration of Neural Networks for Characterizing Voids in Welded Strips	455
<i>Matteo Cacciola, Salvatore Calcagno, Filippo Laganá, Giuseppe Megali, Diego Pellicanó, Mario Versaci, and Francesco Carlo Morabito</i>	
Connectionist Models for Formal Knowledge Adaptation	465
<i>Ilianna Kollia, Nikolaos Simou, Giorgos Stamou, and Andreas Stafylopatis</i>	
Modeling Human Operator Controlling Process in Different Environments	475
<i>Darko Kovacevic, Nikica Pribacic, Mate Jovic, Radovan Antonic, and Asja Kovacevic</i>	
Discriminating between V and N Beats from ECGs Introducing an Integrated Reduced Representation along with a Neural Network Classifier	485
<i>Vaclav Chudacek, George Georgoulas, Michal Huptych, Chrysostomos Stylios, and Lenka Lhotska</i>	
Mental Tasks Classification for a Noninvasive BCI Application	495
<i>Alexandre Ormiga G. Barbosa, David Ronald A. Diaz, Marley Maria B.R. Vellasco, Marco Antonio Meggiolaro, and Ricardo Tanscheit</i>	
Municipal Creditworthiness Modelling by Radial Basis Function Neural Networks and Sensitive Analysis of Their Input Parameters	505
<i>Vladimir Olej and Petr Hajek</i>	
A Comparison of Three Methods with Implicit Features for Automatic Identification of P300s in a BCI	515
<i>Luigi Sportiello, Bernardo Dal Seno, and Matteo Matteucci</i>	
Neural Dynamics and Complex Systems	
Computing with Probabilistic Cellular Automata	525
<i>Martin Schüle, Thomas Ott, and Ruedi Stoop</i>	
Delay-Induced Hopf Bifurcation and Periodic Solution in a BAM Network with Two Delays	534
<i>Jian Xu, Kwok Wai Chung, Ju Hong Ge, and Yu Huang</i>	

Response Properties to Inputs of Memory Pattern Fragments in Three Types of Chaotic Neural Network Models	544
<i>Hamada Toshiyuki, Jousuke Kuroiwa, Hisakazu Ogura, Tomohiro Odaka, Haruhiko Shirai, and Yuko Kato</i>	
Partial Differential Equations Numerical Modeling Using Dynamic Neural Networks	552
<i>Rita Fuentes, Alexander Poznyak, Isaac Chairez, and Tatyana Poznyak</i>	
The Lin-Kernighan Algorithm Driven by Chaotic Neurodynamics for Large Scale Traveling Salesman Problems	563
<i>Shun Motohashi, Takafumi Matsuura, Tohru Ikeguchi, and Kazuyuki Aihara</i>	
Quadratic Assignment Problems for Chaotic Neural Networks with Dynamical Noise	573
<i>Takayuki Suzuki, Shun Motohashi, Takafumi Matsuura, Tohru Ikeguchi, and Kazuyuki Aihara</i>	
Global Exponential Stability of Recurrent Neural Networks with Time-Dependent Switching Dynamics	583
<i>Zhilang Zeng, Jun Wang, and Tingwen Huang</i>	
Approximation Capability of Continuous Time Recurrent Neural Networks for Non-autonomous Dynamical Systems	593
<i>Yuichi Nakamura and Masahiro Nakagawa</i>	
Spectra of the Spike Flow Graphs of Recurrent Neural Networks	603
<i>Filip Piekarniewski</i>	
Activation Dynamics in Excitable Maps: Limits to Communication Can Facilitate the Spread of Activity	613
<i>Andreas Loengarov and Valery Tereshko</i>	
Vision and Image Processing	
Learning Features by Contrasting Natural Images with Noise	623
<i>Michael Gutmann and Aapo Hyvärinen</i>	
Feature Selection for Neural-Network Based No-Reference Video Quality Assessment	633
<i>Dubravko Ćulibrk, Dragan Kukolj, Petar Vasiljević, Maja Pokrić, and Vladimir Zlokolica</i>	
Learning from Examples to Generalize over Pose and Illumination.....	643
<i>Marco K. Müller and Rolf P. Würtz</i>	

Semi-supervised Learning with Constraints for Multi-view Object Recognition	653
<i>Stefano Melacci, Marco Maggini, and Marco Gori</i>	
Large-Scale Real-Time Object Identification Based on Analytic Features	663
<i>Stephan Hasler, Heiko Wersing, Stephan Kirstein, and Edgar Körner</i>	
Estimation Method of Motion Fields from Images by Model Inclusive Learning of Neural Networks	673
<i>Yasuaki Kuroe and Hajimu Kawakami</i>	
Hybrid Neural Systems for Reduced-Reference Image Quality Assessment	684
<i>Judith Redi, Paolo Gastaldo, and Rodolfo Zunino</i>	
Representing Images with χ^2 Distance Based Histograms of SIFT Descriptors	694
<i>Ville Viitaniemi and Jorma Laaksonen</i>	
Modelling Image Complexity by Independent Component Analysis, with Application to Content-Based Image Retrieval	704
<i>Jukka Perkiö and Aapo Hyvärinen</i>	
Adaptable Neural Networks for Objects' Tracking Re-initialization	715
<i>Anastasios Doulamis</i>	
Lattice Independent Component Analysis for fMRI Analysis	725
<i>Manuel Graña, Maite García-Sebastián, and Carmen Hernández</i>	
Adaptive Feature Transformation for Image Data from Non-stationary Processes	735
<i>Erik Schaffernicht, Volker Stephan, and Horst-Michael Gross</i>	
Bio-inspired Connectionist Architecture for Visual Detection and Refinement of Shapes	745
<i>Pedro L. Sánchez Orellana and Claudio Castellanos Sánchez</i>	
Neuro-Evolution and Hybrid Techniques for Mobile Agents Control	
Evolving Memory Cell Structures for Sequence Learning	755
<i>Justin Bayer, Daan Wierstra, Julian Togelius, and Jürgen Schmidhuber</i>	
Measuring and Optimizing Behavioral Complexity for Evolutionary Reinforcement Learning	765
<i>Faustino J. Gomez, Julian Togelius, and Juergen Schmidhuber</i>	
Combining Multiple Inputs in HyperNEAT Mobile Agent Controller	775
<i>Jan Drchal, Ondrej Kapral, Jan Koutník, and Miroslav Šnorek</i>	

Evolving Spiking Neural Parameters for Behavioral Sequences	784
<i>Thomas M. Poulsen and Roger K. Moore</i>	
Robospike Sensory Processing for a Mobile Robot Using Spiking Neural Networks	794
<i>Michael F. McBride, T.M. McGinnity, and Liam P. Maguire</i>	
Neural Control, Planning and Robotics Applications	
Basis Decomposition of Motion Trajectories Using Spatio-temporal NMF	804
<i>Sven Hellbach, Julian P. Eggert, Edgar Körner, and Horst-Michael Gross</i>	
An Adaptive NN Controller with Second Order SMC-Based NN Weight Update Law for Asymptotic Tracking	815
<i>Haris Psillakis</i>	
Optimizing Control by Robustly Feasible Model Predictive Control and Application to Drinking Water Distribution Systems	823
<i>Vu Nam Tran and Mietek A. Brdys</i>	
Distributed Control over Networks Using Smoothing Techniques	835
<i>Ion Necoara</i>	
Trajectory Tracking of a Nonholonomic Mobile Robot Considering the Actuator Dynamics: Design of a Neural Dynamic Controller Based on Sliding Mode Theory	845
<i>Nardênia A. Martins, Douglas W. Bertol, and Edson R. De Pieri</i>	
Tracking with Multiple Prediction Models	855
<i>Chen Zhang and Julian Eggert</i>	
Sliding Mode Control for Trajectory Tracking Problem - Performance Evaluation	865
<i>Razvan Solea and Daniela Cernega</i>	
Bilinear Adaptive Parameter Estimation in Fuzzy Cognitive Networks	875
<i>Thodoris Kottas, Yiannis Boutalis, and Manolis Christodoulou</i>	
Intelligent Tools and Methods for Multimedia Annotation	
AM-FM Texture Image Analysis of the Intima and Media Layers of the Carotid Artery	885
<i>Christos P. Loizou, Victor Murray, Marios S. Pattichis, Christodoulos S. Christodoulou, Marios Pantziaris, Andrew Nicolaides, and Constantinos S. Pattichis</i>	

Unsupervised Clustering of Clickthrough Data for Automatic Annotation of Multimedia Content	895
<i>Klimis Ntalianis, Anastasios Doulamis, Nicolas Tsapatsoulis, and Nikolaos Doulamis</i>	
Object Classification Using the MPEG-7 Visual Descriptors: An Experimental Evaluation Using State of the Art Data Classifiers	905
<i>Nicolas Tsapatsoulis and Zenonas Theodosiou</i>	
MuLVAT: A Video Annotation Tool Based on XML-Dictionaries and Shot Clustering	913
<i>Zenonas Theodosiou, Anastasis Kounoudes, Nicolas Tsapatsoulis, and Marios Milis</i>	
Multimodal Sparse Features for Object Detection	923
<i>Martin Haker, Thomas Martinetz, and Erhardt Barth</i>	
Critical Infrastructure Systems	
Multiple Kernel Learning of Environmental Data. Case Study: Analysis and Mapping of Wind Fields	933
<i>Loris Foresti, Devis Tuia, Alexei Pozdnoukhov, and Mikhail Kanevski</i>	
Contributor Diagnostics for Anomaly Detection	944
<i>Alexander Borisov, George Runger, and Eugene Tuv</i>	
Indoor Localization Using Neural Networks with Location Fingerprints	954
<i>Christos Laoudias, Demetrios G. Eliades, Paul Kemppi, Christos G. Panayiotou, and Marios M. Polycarpou</i>	
Distributed Faulty Sensor Detection in Sensor Networks	964
<i>Xuanwen Luo and Ming Dong</i>	
Detection of Failures in Civil Structures Using Artificial Neural Networks	976
<i>Zhan Wei Lim, Colin Keng-Yan Tan, Winston Khoon-Guan Seah, and Guan-Hong Tan</i>	
Congestion Control in Autonomous Decentralized Networks Based on the Lotka-Volterra Competition Model	986
<i>Pavlos Antoniou and Andreas Pitsillides</i>	
Author Index	997

Mutual Information Based Initialization of Forward-Backward Search for Feature Selection in Regression Problems

Alberto Guillén¹, Antti Sorjamaa², Gines Rubio³,
Amaury Lendasse², and Ignacio Rojas³

¹ Department of Informatics
University of Jaen, Spain

² Department of Computer Architecture and Technology
University of Granada, Spain

³ Department of Information and Computer Science
Helsinki University of Technology, Finland

Abstract. Pure feature selection, where variables are chosen or not to be in the training data set, still remains as an unsolved problem, especially when the dimensionality is high. Recently, the Forward-Backward Search algorithm using the Delta Test to evaluate a possible solution was presented, showing a good performance. However, due to the locality of the search procedure, the initial starting point of the search becomes crucial in order to obtain good results. This paper presents new heuristics to find a more adequate starting point that could lead to a better solution. The heuristic is based on the sorting of the variables using the Mutual Information criterion, and then performing parallel local searches. These local searches provide an initial starting point for the actual parallel Forward-Backward algorithm.

1 Introduction

Input selection is a crucial part when building an approximator. Too many input variables increase the calculation time and model complexity and even lead to suboptimal results. On the other hand too few variables might not contain all the relevant information for an accurate approximation.

In many cases, the approximator cannot be used to test all possible combinations of variables in order to find the optimal one. That can be due to the huge number of combinations, which increases exponentially with respect to the number of variables, or due to the fact that not all approximators can distinguish the relevant inputs from the bogus ones.

In this paper, a greedy selection methodology, called Forward-Backward Search, is used to select the variables. It relies on the Delta Test estimation methodology [1]. Even though Forward-Backward is not going through all possible solutions and does not guarantee the optimality of the final selection, it always finds a local optimal one.

Because of the locality of the Forward-Backward, a good initialization is crucial, and this paper presents several new heuristics for the initialization. The Forward-Backward methodology is also deterministic with respect to the initialization; the same initial selection of variables provides the same final solution.

The rest of the paper is organized as follows: Section 2 presents the Forward-Backward Search algorithm and the theoretical background of the Delta Test. Then, Section 3 introduces the new improvements incorporated to enhance the variable selection. Afterwards, Section 4 shows an experimental result, where the heuristics are briefly compared.

2 Forward-Backward Search

Forward-Backward Search (FBS) is an algorithm that results from the joining of two methodologies: Forward and Backward selections [2]. Both the Forward Selection and the Backward Elimination (or Pruning) methods suffer from an incomplete search. The FBS offers the flexibility to reconsider input variables previously discarded and *vice versa*, to discard input variables previously selected. It can start from any initial input set, including empty, full or randomly initialized input set.

Let us suppose a set of inputs X^i , $i = 1, 2, \dots, d$ and output Y , the procedure of the Forward-Backward Search is summarized in Figure 1. In the procedure example the k -Nearest Neighbors (k NN) criterion [3] is used as an example criterion for evaluating the input set, but the criterion can be almost any criteria or a suitable approximator.

1. (Initialization)

Let set S be the selected input set, which can contain any input variables, and set F be the unselected input set containing the inputs, which are not in set S . Compute k NN(S, Y) error.

2. (Forward-Backward Search)

Find:

$$X^s = \arg \max_{X^i, X^j} \{k\text{NN}(\{S, X^j\}, Y) \cup \{k\text{NN}(S \setminus X^i, Y)\}, X^i \in S, X^j \in F.$$

If the previously computed k NN error is smaller than the new k NN error, stop; otherwise, update set S and save the new k NN error, repeat step 2 until S is equal to any former selected S .

3. (Result)

The selection result is in set S .

Fig. 1. Forward-Backward Search Strategy

It is noted that the selection result depends on the initialization of the input set. In this paper, several options are considered and the options are discussed more deeply in Section 3.

In the course of FBS procedure, the number of evaluated input sets varies and is dependent on the initialization of the input set, the stopping criteria and the nature of the problem. Still, it is not guaranteed that in all cases this selection method finds the global optimal input set.

2.1 The Delta Test

The Delta Test (DT), introduced by Pi and Peterson for time series [4] and proposed for variable selection in [1], is a technique to estimate the variance of the noise, or the mean squared error (MSE), that can be achieved without overfitting. Given N input-output pairs $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, the relationship between \mathbf{x}_i and y_i can be expressed as

$$y_i = f(\mathbf{x}_i) + r_i, \quad i = 1, \dots, N \quad (1)$$

where f is an unknown function and r is the noise. The DT estimates the variance of the noise r .

The DT is useful for evaluating the nonlinear correlation between two random variables, namely, input and output pairs. The DT can also be applied to input variable selection: the set of input variables that minimizes the DT is the one that is selected. Indeed, according to the DT, the selected set of input variables is the one that represents the relationship between input variables and the output variable in the most deterministic way.

The DT is based on a hypothesis coming from the continuity of the regression function. If two points \mathbf{x} and \mathbf{x}' are close in the input space, the continuity of the regression function implies that the outputs $f(\mathbf{x})$ and $f(\mathbf{x}')$ are also close enough in the output space. Alternatively, if the corresponding output values are not close in the output space, this is due to the influence of the noise.

The DT can be interpreted as a particularization of the Gamma Test [5] considering only the first nearest neighbor. Let us denote the first nearest neighbor of a point \mathbf{x}_i in the \mathbb{R}^d space as $\mathbf{x}_{NN(i)}$. The nearest neighbor formulation of the DT estimates $\text{Var}[r]$ by

$$\text{Var}[r] \approx \delta = \frac{1}{2N} \sum_{i=1}^N (y_i - y_{NN(i)})^2, \text{ with } \text{Var}[\delta] \rightarrow 0 \text{ for } N \rightarrow \infty \quad (2)$$

where $y_{NN(i)}$ is the output of $\mathbf{x}_{NN(i)}$.

3 New Initialization Heuristics for the Forward-Backward Search

The following subsections describe the different approaches proposed to define an adequate starting point. The original FBS has been parallelized in order to take advantage of the architectures available nowadays. The search for the best solution is distributed to several computers in order to have more solutions in less time.

The parallel implementation is quite straightforward and consists of the division of the generated subsets of variables from the first iteration of the FBS. This division is shown in Figure 2.

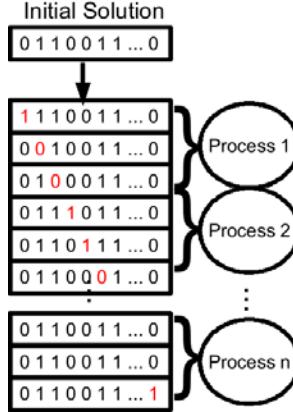


Fig. 2. Parallel scheme for the Forward-Backward Search

Once each process has a part of the subset, they proceed as the original FBS. The algorithm stops when all the processes have converged to a solution. Then, the best solution is found among the final solutions of the individual processes.

3.1 Simple Mutual Information Based Initialization

Let $X^l = \{x_m^l\}$ with $l \in 1, \dots, d$ (i.e. X^l is the l -th input variable) and $Y = \{y_m\}$ with $\{m = 1 \dots M\}$. The Mutual Information (MI) between X^l and Y can be defined as the amount of information that X^l provides about Y , and can be expressed as:

$$I(X^l, Y) = \sum_{y \in Y} \sum_{x \in X} \mu_{X^l, Y}(x, y) \log \frac{\mu_{X^l, Y}(x, y)}{\mu_X^l(x) \mu_Y(y)}. \quad (3)$$

$\mu_{X^l, Y}$ is the joint probability distribution function of X^l and Y , and $\mu_X^l(x)$ and $\mu_Y(y)$ are the marginal probability distribution functions of X^l and Y respectively.

Therefore, in order to obtain the MI between X^l and Y , only the estimate of the joint probability density function is needed. This value can be computed using several techniques based on histograms, kernels or the k NN. In this paper, the one based on the k NN is used [6].

For each input variable, the MI between that variable and the output is computed and, once finished, it is possible to rank all the input variables according the values of MI. Then, the initial solution for the FBS is defined as a number

of *first* variables in the ranking. The problem now is to determine the actual number of variables, since the value obtained by the MI is not enough to perform this selection. Unfortunately, the only chance is to set this value manually.

Another issue is to determine the value of k for the k NN algorithm that computes the MI. Although there is a possibility of guessing an adequate value: compute the MI values using several k and select the one that provides the highest values.

3.2 RaVI: Ranked Variables Initialization

Being aware of the two significant drawbacks of the used MI heuristic (the determination of the k and the final number of variables to be chosen), another heuristic is considered. This new heuristic requires the definition of only one parameter. This value can again be set manually as in the MI, or as a function of the available computational resources making the heuristic more flexible when executed in different computer architectures or systems.

The heuristic works as follows: it performs a division of the original input vector into subvectors of a smaller dimension. Then, the local search is applied to each subvector.

When the search is focused on the subvector, there are several possibilities in handling the rest of the inputs. In this paper, we are considering four starting alternatives: 1) all zeros, 2) subvectors ones, the other inputs zeros, 3) all ones, 4) subvectors zeros, the other inputs ones.

The RaVI scheme is summarized in Figure 3.

In the Figure, the initial solutions are divided into slices of size 3 (depicted as x) and the remaining values are not changed (depicted as -) during the first FBS. This first FBS is done sequentially and separately in each process. Once all the processes have converged to a local optima, the processes perform a collective communication and share the results found by the other processes. Then, the initial starting point for the parallel FBS (named as middle solution in Figure 3) is computed by concatenating all local solutions. Finally, the parallel FBS, presented in Figure 2, is performed starting from the middle solution.

The new aspect of this method is the sorting of the variables before performing the slice division and the following local searches. Since the variables are going to be analyzed with their neighbors in a local manner, it might be convenient to rank the variables with high MI values close to each other when performing the local searches. With the MI, in this case, it is not so crucial to select the k for the MI as the most optimal one, because each of the slices will contain similar variables in terms of their MI values. When the local results are concatenated, the middle solution includes inputs with a wide range of MI values, only the optimality inside each slice is emphasized.

Here, we use two different sorting schemes based on the MI criterion. The first one sorts the variables in a descending order, starting from the variable with the largest MI value and ending up with the one with the lowest.

The second sorting scheme, called RaVI Mix, is aiming to bunch together variables with high and low values of MI. This sorting gives more chances to the

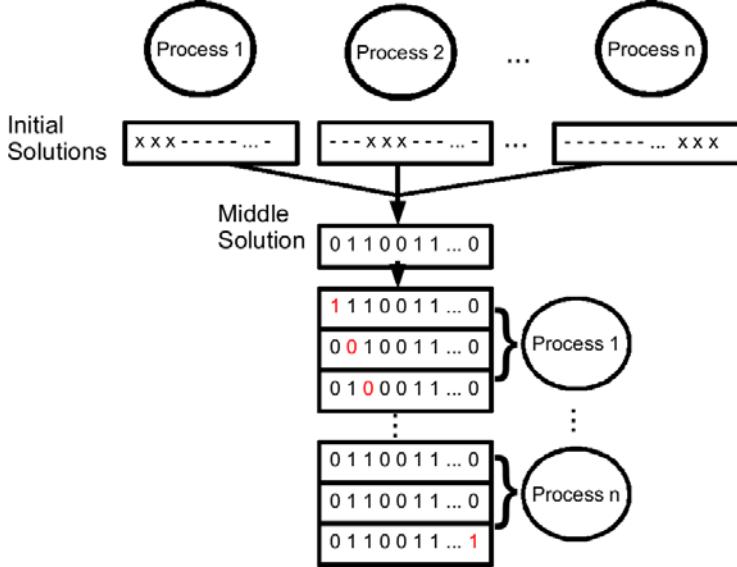


Fig. 3. Scheme of the algorithm using the slice division heuristic to initialize the starting point for the FBS

sublocal searches to select good variables, whether they have large MI value or not. The second sorting scheme is visualized in the following:

$$\left[X^{\text{MI}(1)} \ X^{\text{MI}(d)} \ X^{\text{MI}(2)} \ X^{\text{MI}(d-1)} \dots X^{\text{MI}(m)} \right], \quad (4)$$

where MI denotes the ranking of all d inputs in the dataset, MI(1) denotes the input with the highest MI value and MI(d) the one with the lowest. m is the middlemost input in the ranking.

After the sorting, the input space is divided into sublocal search spaces, or slices, as demonstrated in Figure 3.

4 Experiments

In this paper, we use a dataset from the recently organized ESTSP 2007 conference. The ESTSP dataset presents a weekly sea temperature for roughly 17 years and contains 874 values and it is shown in Figure 4.

The dataset is transformed into a regressor of 55 input variables, and one output variable, and a total of 819 samples using a sliding window over the whole dataset. We have a time series prediction problem of one step ahead and we want to do the input variable selection in order to decrease the amount of input variables. All the samples are used in the variable selection part.

We used 8 processes (or processors) with each method, which set the sizes of the slices to 6 or 7 (7 slices with 7 variables and one with 6 variables). As already mentioned, the number of processes can be defined manually or with respect to the

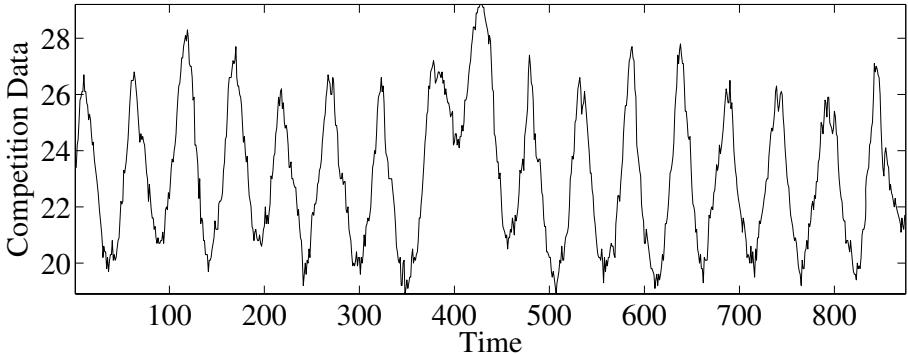


Fig. 4. The ESTSP 2007 Competition dataset

computational resources available. The parallel implementation was performed using Matlab software and recently developed MPI implementation [7].

In the preliminary tests performed before the actual comparison of the methods, a well-known Housing dataset was used to verify the correctness of the methods. Because Housing dataset includes only 13 inputs, it is possible to use exhaustive search to compute the global optimum according to the Delta Test. Using any of the presented heuristics with any presented initialization, the global optimum was always found.

Table 1 summarizes the results of all methods using the ESTSP 2007 Competition dataset.

Table 1. Results of all methods using the ESTSP 2007 Competition dataset

Heuristic	Starting Point	# Variables	Delta Test	Solutions	Time
pFBS	All	46	0.0284	3856	248
	None	18	0.0299	4640	168
Simple MI	10 best MI	15	0.0272	8056	222
RaVI	All	22	0.0269	4243	194
	None	28	0.0277	6629	282
	Ones and Zeros	19	0.0267	3541	145
	Zeros and Ones	31	0.0283	5398	236
RaVI Mix	All	21	0.0284	3755	156
	None	32	0.0264	5445	289
	Ones and Zeros	25	0.0269	4077	148
	Zeros and Ones	28	0.0293	6358	273

From Table 1 we can see that the RaVI Mix methodology obtains the lowest Delta Test value and, therefore, has selected the best set of inputs. However, there are no big differences among the methods, even though the Delta Test

value with all variables is 1.1765. It means that each of the presented heuristics have done their job adequately.

The Exhaustive Search of all possible combinations in the 55 dimensional space is clearly unfeasible task and one must use some sort of heuristics to ease the search process. The Table I shows that the MI alone is not enough to guide the FBS toward more optimal selection of input variables. Even though the Simple MI heuristic searched through the largest amount of input combinations, it was not able to find better solution than RaVI Mix heuristic.

Furthermore, we observed that although there is a possibility to end up in the same solutions when using the parallel implementation of the FBS, only few searches ended up with the same local minima. Roughly only one percent of the solutions searched through were already evaluated by another parallel computation thread.

Figure 5 shows the selected variables using the RaVI Mix selection scheme and some inputs with high and low MI values for comparison.

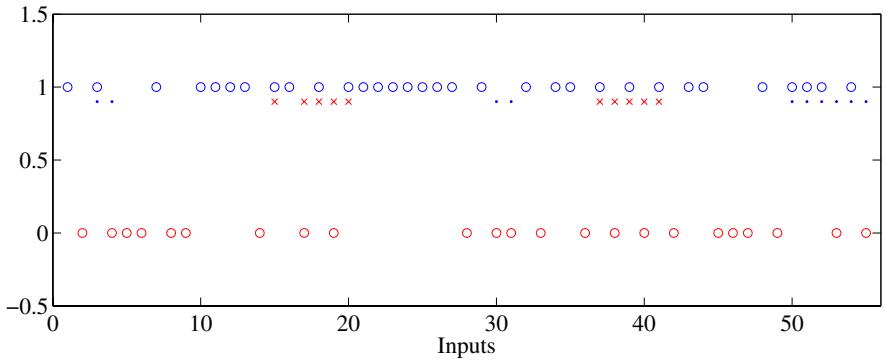


Fig. 5. Selected inputs using the RaVI Mix method starting from all zeros. The input variables are on the horizontal axis, red circles at zero depict not selected variables and blue circles at one depict selected variables. Blue dots at 0.9 denote the variables with 10 highest MI values and red crosses the ones with the 10 lowest values.

From Figure 5 we can see that the selection is not selecting all variables with the highest MI values, but also the ones with very low value. For example, variables 30 and 31 are among the variables with the highest MI values, but none of them is selected. On the other hand, variables from 37 to 41 are among the variables with the lowest MI values, but some of them are chosen by the RaVI Mix.

This suggests that the MI value alone is not able to give a clear justification to use the variable, and that also the variables with low MI value can be useful in the approximation.

5 Conclusions

The problem of finding a good subset of variables for any kind of regression model is still remaining as an unsolved problem. Due to the high dimensionality of the real-life problems, it is not possible to apply an exhaustive search that would provide the global optimum.

Within this context, this paper presents several heuristics to improve the behavior of a previously published algorithm, the Forward-Backward Search. These new heuristics rely on the theoretical basis provided by the Mutual Information. The search starts from a point that could be closer to adequate local minimum.

Another relevant aspect of one of the heuristics is the possibility to analyze the relationships between variables, defining neighbor relationships. This aspect can be further studied using different metrics, since this last heuristic provided the best results in a complex input selection problem.

For further work, other ranking criteria will be tested and the effect of using different sizes of local searches will be quantified and compared. Also other means of local estimation of a good selection of inputs will be tried.

Acknowledgment

This work has been partially supported by the projects TIN2007-60587, P07-TIC-02768 and P07-TIC-02906, TIC-3928. Part of the work of Antti Sorjamaa is supported by a grant from Nokia Foundation, Finland.

References

1. Eirola, E., Liitiäinen, E., Lendasse, A., Corona, F., Verleysen, M.: Using the Delta Test for Variable Selection. In: ESANN 2008, European Symposium on Artificial Neural Networks, Bruges, Belgium (April 2008)
2. Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., Lendasse, A.: Methodology for long-term prediction of time series. *Neurocomputing* 70(16-18), 2861–2869 (2007)
3. Bishop, C.: *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford (1995)
4. Pi, H., Peterson, C.: Finding the embedding dimension and variable dependencies in time series. *Neural Computation* 6(3), 509–520 (1994)
5. Jones, A.J.: New tools in non-linear modelling and prediction. *Computational Management Science* 1(2), 109–149 (2004)
6. Kraskov, A., Stögbauer, H., Grassberger, P.: Estimating mutual information. *Phys. Rev.* 69, 66–138 (2004)
7. Guillen, A., Rojas, I., Rubio, G., Pomares, H., Herrera, L.J., Gonzalez, J.: A new interface for MPI in matlab and its application over a genetic algorithm. In: Lendasse, A. (ed.) *Proceedings of the European Symposium on Time Series Prediction*, pp. 37–46 (2008), <http://atc.ugr.es/~aguillen>

Kernel Learning for Local Learning Based Clustering

Hong Zeng and Yiu-ming Cheung*

Department of Computer Science
Hong Kong Baptist University, Hong Kong SAR, China
{hzeng, ymc}@comp.hkbu.edu.hk

Abstract. For most kernel-based clustering algorithms, their performance will heavily hinge on the choice of kernel. In this paper, we propose a novel kernel learning algorithm within the framework of the Local Learning based Clustering (LLC) (Wu & Schölkopf 2006). Given multiple kernels, we associate a non-negative weight with each Hilbert space for the corresponding kernel, and then extend our previous work on feature selection (Zeng & Cheung 2009) to select the suitable Hilbert spaces for LLC. We show that it naturally renders a linear combination of kernels. Accordingly, the kernel weights are estimated iteratively with the local learning based clustering. The experimental results demonstrate the effectiveness of the proposed algorithm on the benchmark document datasets.

1 Introduction

In the past few decades, the kernel methods have been widely applied to various learning problems, where the data is implicitly mapped into a nonlinear high dimensional space by kernel function [3]. Unfortunately, it is known that the performance heavily hinges on the choice of kernel, and the most suitable kernel for a particular task is often unknown in advance. Thereby, learning an appropriate kernel, is critical to obtain an improved performance for the employed kernel-based inference method.

In this paper, we are particularly interested in the problem of kernel learning for clustering. In the literature, the kernel learning has been extensively studied for the supervised learning contexts. However, this issue remains less explored in unsupervised problems, due to the absence of ground truth class labels that could guide the learning for “ideal” kernels. Until very recently, several algorithms have been proposed to address this issue for clustering. Some approaches [4][5] directly learn the kernel parameters of some specific kernels. Though improvement is often achieved, extension of the learning method to other kernel functions is often nontrivial. A more effective framework, termed as the multiple kernel learning [6], learns a linear combination of base kernels with different weights, which will be estimated iteratively with the inference process [7][8]. This strategy may bring potential advantages over those which try to obtain a single best kernel, through exploiting the complementary information among different kernels. In [7], the algorithm tries to find a maximum margin hyperplane to

* Yiu-ming Cheung (ymc@comp.hkbu.edu.hk) is the corresponding author. This work was supported by the Faculty Research Grant of HKBU under Project: FRG/07-08/II-54, and the Research Grant Council of Hong Kong SAR under Grant: HKBU 210306.

cluster data (restricted to binary-class case), accompanied with learning a mixture of Laplacian matrices. In [8], clustering is phrased as a non-negative matrix factorization problem of a fused kernel matrices. Nevertheless, both approaches in [7,8] are global learning based. Their performance may be degraded when samples are less separable from a global view.

Under the circumstances, we therefore propose a novel multiple kernel learning method within the framework of the Local Learning based Clustering (LLC) [1], which aims at optimizing the local purity requirement of clustering assignment. It is expected that it will produce a more reliable intermediate clustering result when the samples are globally less separable. We associate a non-negative weight with each Hilbert space (or called the feature space interchangeably) for the corresponding kernel, and then extend our previous work on feature selection [2] to select the suitable Hilbert spaces for LLC. Such strategy naturally leads to learn a linear combination of all the available kernels at hand. Accordingly, an algorithm is developed in which the combination coefficients of kernels are estimated iteratively with the local learning based clustering.

The remainder of the paper is organized as follows: Section 2 gives an overview of local learning based clustering algorithm. We present the proposed method in Section 3. In Section 4, the experiments on several benchmark datasets are presented. We draw a conclusion in Section 5.

2 Overview of the Local Learning Based Clustering Algorithm

Let us first introduce the indicator matrix that will be used later. Suppose n data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n (\mathbf{x}_i \in \mathbb{R}^d)$ will be partitioned into C clusters. The clustering result can be represented by a *cluster assignment indicator matrix* $\mathbf{P} = [p_{ic}] \in \{0, 1\}^{n \times C}$, such that $p_{ic} = 1$ if \mathbf{x}_i belongs to the c th cluster, and $p_{ic} = 0$ otherwise. The *scaled cluster assignment indicator matrix* used in this paper is defined by: $\mathbf{Y} = \mathbf{P}(\mathbf{P}^T \mathbf{P})^{-\frac{1}{2}} = [\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^C]$, where $\mathbf{y}^c = [y_{1c}, \dots, y_{nc}]^T \in \mathbb{R}^n (1 \leq c \leq C)$, is the c -th column of $\mathbf{Y} \in \mathbb{R}^{n \times C}$. $y_{ic} = p_{ic}/\sqrt{n_c}$ can be regarded as the confidence that \mathbf{x}_i is assigned to the c th cluster, where n_c is the size of the c th cluster. It is easy to verify that $\mathbf{Y}^T \mathbf{Y} = \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix.

The starting point of the LLC [1] is that the cluster assignments in the neighborhood of each point should be as pure as possible. Suppose there exists an arbitrary \mathbf{Y} at first, for each \mathbf{x}_i , a regression model is built with the training data $\{(\mathbf{x}_j, y_{jc})\}_{\mathbf{x}_j \in \mathcal{N}_i} (1 \leq c \leq C, 1 \leq i, j \leq n)$, where \mathcal{N}_i denotes the set of neighboring¹ points of \mathbf{x}_i (not including \mathbf{x}_i itself). The output of the local model is of the following form: $f_i^c(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\theta}_i^c, \forall \mathbf{x} \in \mathbb{R}^d$, where $\boldsymbol{\theta}_i^c \in \mathbb{R}^d$ is the local regression coefficients vector. Here, the bias term is ignored for simplicity, provided that one of the features is 1. In [1], $\boldsymbol{\theta}_i^c$ is solved by:

$$\min_{\boldsymbol{\theta}_i^c} \sum_{c=1}^C \sum_{i=1}^n \left[\sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta(y_{jc} - \mathbf{x}_j^T \boldsymbol{\theta}_i^c)^2 + \|\boldsymbol{\theta}_i^c\|^2 \right], \quad (1)$$

¹ The k -mutual neighbors are adopted in order to well describe the local structure, i.e. \mathbf{x}_j is considered as a neighbor of \mathbf{x}_i only if \mathbf{x}_i is also one of the k -nearest neighbors of \mathbf{x}_j .

where β is a trade-off parameter. Denote the solution to the linear ridge regression problem ① as θ_i^{c*} , the predicted cluster assignment for the test data \mathbf{x}_i can then be calculated by: $\hat{y}_{ic} = f_i^c(\mathbf{x}_i) = \mathbf{x}_i^T \theta_i^{c*} = \alpha_i^T \mathbf{y}_i^c$, where

$$\alpha_i^T = \beta \mathbf{x}_i^T (\beta \mathbf{X}_i \mathbf{X}_i^T + \mathbf{I})^{-1} \mathbf{X}_i, \quad (2)$$

$\mathbf{X}_i = [\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_{n_i}}]$ with \mathbf{x}_{i_k} being the k -th neighbor of \mathbf{x}_i , n_i is the size of \mathcal{N}_i , and $\mathbf{y}_i^c = [y_{i_1c}, y_{i_2c}, \dots, y_{i_{n_i}c}]^T$.

After all the local predictors have been constructed, LLC aims to find an optimal cluster indicator matrix \mathbf{Y} via minimizing the overall prediction errors:

$$\sum_{c=1}^C \sum_{i=1}^n (y_{ic} - \hat{y}_{ic})^2 = \sum_{c=1}^C \|\mathbf{y}^c - \mathbf{A}\mathbf{y}^c\|^2 = \text{trace}(\mathbf{Y}^T \mathbf{T} \mathbf{Y}), \quad (3)$$

where $\mathbf{T} = (\mathbf{I} - \mathbf{A})^T (\mathbf{I} - \mathbf{A})$, \mathbf{A} is an $n \times n$ sparse matrix with its (i, j) -th entry a_{ij} being the corresponding element in α_i by ② if $\mathbf{x}_j \in \mathcal{N}_i$ and 0 otherwise.

As in the spectral clustering [9][10], \mathbf{Y} is relaxed into the continuous domain while keeping the property $\mathbf{Y}^T \mathbf{Y} = \mathbf{I}$ for ③. LLC then solves:

$$\min_{\mathbf{Y} \in \mathbb{R}^{n \times C}} \text{trace}(\mathbf{Y}^T \mathbf{T} \mathbf{Y}) \text{ s.t. } \mathbf{Y}^T \mathbf{Y} = \mathbf{I} \quad (4)$$

A solution to \mathbf{Y} is given by the first C eigenvectors of the matrix \mathbf{T} , corresponding to the first C smallest eigenvalues. The final partition result is obtained by discretizing \mathbf{Y} via the method in [10] or by k-means as in [9].

3 Multiple Kernel Learning for Local Learning Based Clustering

The LLC algorithm can be easily kernelized as in [1], by replacing the linear ridge regression with the kernel ridge regression. Under the circumstances, selecting a suitable kernel function will be a crucial issue. We extend our previous work of feature selection for LLC [2] to learn a proper linear combination of several pre-computed kernel matrices.

In the kernel methods, the symmetric positive semi-definite kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, implicitly maps the raw input features into a high-dimensional (possibly infinite) *Reproducing Kernel Hilbert Space* (RKHS) \mathcal{H} , which is equipped with the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ via a nonlinear mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$, i.e., $\mathcal{K}(x, z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{H}}$. Suppose there are L different available kernel functions $\{\mathcal{K}^{(l)}\}_{l=1}^L$. Accordingly, there are L different associated feature spaces $\{\mathcal{H}^{(l)}\}_{l=1}^L$. Since it is unknown which feature space should be used, an intuitive way is to use them all by concatenating all feature spaces into an augmented Hilbert space: $\tilde{\mathcal{H}} = \bigoplus_{l=1}^L \mathcal{H}^{(l)}$, and associate each feature space with a relevance weight τ_l ($\sum_{l=1}^L \tau_l = 1, \tau_l \geq 0, \forall l$), or equivalently the importance factor for kernel function $\mathcal{K}^{(l)}$. Later, we will show that performing LLC in such feature space is equivalent to employing a combined kernel function: $\mathcal{K}^{\tau}(x, z) = \sum_{l=1}^L \tau_l \mathcal{K}^{(l)}(x, z)$ for LLC. A zero weight τ_l will correspond to *blend out* the feature space associated with the corresponding kernel similar to the feature selection in [2]. Our task is to learn the coefficients $\{\tau_l\}_{l=1}^L$ which can lead to a more accurate and robust performance. Subsequently, an algorithm that iteratively performs clustering and estimates the kernel weight is developed.

3.1 Update Y for a Given τ

First of all, given a τ , the nearest neighbors \mathcal{N}_i for LLC algorithm will be re-found by the τ -weighted squared Euclidean distance in $\tilde{\mathcal{H}}$, i.e.:

$$d_\tau(\mathbf{x}_1, \mathbf{x}_2) = \|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)\|_\tau^2 = \mathcal{K}^\tau(\mathbf{x}_1, \mathbf{x}_1) + \mathcal{K}^\tau(\mathbf{x}_2, \mathbf{x}_2) - 2\mathcal{K}^\tau(\mathbf{x}_1, \mathbf{x}_2). \quad (5)$$

Then the local discriminant function in the $\tilde{\mathcal{H}}$ can be written as follows:

$$f_i^c(\phi(\mathbf{x})) = \phi(\mathbf{x})^T \mathbf{w}_i^c + b_i^c, \quad (6)$$

where $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}) \ \phi_2(\mathbf{x}) \ \cdots \ \phi_L(\mathbf{x})]^T \in \mathbb{R}^D$, $\phi_l(\mathbf{x}) \in \mathbb{R}^{D_l}$ is the sample mapped by the l th kernel function, $\sum_{l=1}^L D_l = D$, D and D_l are the dimensionalities of $\tilde{\mathcal{H}}$ and $\mathcal{H}^{(l)}$, respectively. Taking the relevance of each feature space for clustering into account, the regression coefficient $\mathbf{w}_i^c \in \mathbb{R}^D$ and the bias $b_i^c \in \mathbb{R}$ now will be solved via the following weighted l_2 norm regularized least square problem:

$$\min_{\mathbf{w}_i^c, b_i^c} \sum_{c=1}^C \sum_{i=1}^n \left[\sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta(y_{jc} - \phi(\mathbf{x}_j)^T \mathbf{w}_i^c - b_i^c)^2 + \mathbf{w}_i^{cT} \mathbf{\Lambda}_\tau^{-1} \mathbf{w}_i^c \right], \quad (7)$$

where $\mathbf{\Lambda}_\tau$ is a diagonal matrix with the vector $\tilde{\boldsymbol{\tau}} = (\underbrace{\tau_1, \dots, \tau_1}_{D_1}, \dots, \underbrace{\tau_L, \dots, \tau_L}_{D_L})^T$ in the diagonal, and $\sum_{l=1}^L \tau_l = 1, \tau_l \geq 0 \ \forall l$. Similar to [2], the weighted l_2 norm (i.e., the second term in the square bracket of (7)) with τ defined on the standard simplex is able to provide adaptive regularization: a large penalty will be imposed on the elements of \mathbf{w}_i^c corresponding to the feature spaces associated with irrelevant kernels. Thus, an improved clustering result can be expected because the vanishing elements in \mathbf{w}_i^c will eliminate the feature spaces with irrelevant kernels from the prediction (c.f. (6)).

After removing the bias term by plugging its optimal solution

$$b_i^c = \frac{1}{n_i} \mathbf{e}_i^T (\mathbf{y}_i^c - \phi(\mathbf{X}_i)^T \mathbf{w}_i^c), \quad (8)$$

into (7), where $\mathbf{e}_i = [1 \ 1 \ \dots \ 1]^T \in \mathbb{R}^{n_i}$, we can reformulate the primal problem (7) as follows:

$$\min_{\mathbf{w}_i^c} \sum_{c=1}^C \sum_{i=1}^n \left[\beta \|\mathbf{\Pi}_i \mathbf{y}_i^c - (\phi(\mathbf{X}_i) \mathbf{\Pi}_i)^T \mathbf{w}_i^c\|^2 + \mathbf{w}_i^{cT} \mathbf{\Lambda}_\tau^{-1} \mathbf{w}_i^c \right], \quad (9)$$

where $\mathbf{\Pi}_i = \mathbf{I}_{n_i} - \frac{1}{n_i} \mathbf{e}_i \mathbf{e}_i^T$ and $\mathbf{I}_{n_i} \in \mathbb{R}^{n_i \times n_i}$ is a unit matrix, $\mathbf{\Pi}_i \mathbf{\Pi}_i = \mathbf{\Pi}_i$. Then we consider the dual formulation of the (9) in terms of \mathbf{w}_i^c . Denote

$$\zeta_i^c = (\phi(\mathbf{X}_i) \mathbf{\Pi}_i)^T \mathbf{w}_i^c - \mathbf{\Pi}_i \mathbf{y}_i^c, \quad (10)$$

then the Lagrangian for problem (9) is

$$\begin{aligned} \mathcal{L}(\{\zeta_i^c, \mathbf{w}_i^c, \gamma_i^c\}) &= \sum_{c=1}^C \sum_{i=1}^n \left(\beta \|\zeta_i^c\|^2 + \mathbf{w}_i^{cT} \mathbf{\Lambda}_\tau^{-1} \mathbf{w}_i^c \right) \\ &\quad - \sum_{c=1}^C \sum_{i=1}^n \gamma_i^{cT} \left((\phi(\mathbf{X}_i) \mathbf{\Pi}_i)^T \mathbf{w}_i^c - \mathbf{\Pi}_i \mathbf{y}_i^c - \zeta_i^c \right), \end{aligned} \quad (11)$$

where γ_i^c 's with $\gamma_i^c \in \mathbb{R}^{n_i}$ are the vectors of Lagrangian dual variables. Taking the derivatives of \mathcal{L} w.r.t. the primal variables ζ_i^c and \mathbf{w}_i^c , and setting them equal to zero, we obtain:

$$\zeta_i^c = -\frac{\gamma_i^c}{2\beta}, \quad \mathbf{w}_i^c = \frac{\Lambda_\tau \phi(\mathbf{X}_i) \Pi_i \gamma_i^c}{2}, \quad (12)$$

and finally we obtain the dual problem:

$$\begin{aligned} \max_{\gamma_i^c} & \sum_{c=1}^C \sum_{i=1}^n -\frac{1}{4\beta} \gamma_i^{cT} \gamma_i^c - \frac{1}{4} \gamma_i^{cT} \Pi_i \phi(\mathbf{X}_i)^T \Lambda_\tau \phi(\mathbf{X}_i) \Pi_i \gamma_i^c + \gamma_i^{cT} \Pi_i \mathbf{y}_i^c = \\ \max_{\gamma_i^c} & \sum_{c=1}^C \sum_{i=1}^n -\frac{1}{4\beta} \gamma_i^{cT} \gamma_i^c - \frac{1}{4} \gamma_i^{cT} \Pi_i \mathbf{K}_i^\tau \Pi_i \gamma_i^c + \gamma_i^{cT} \Pi_i \mathbf{y}_i^c. \end{aligned} \quad (13)$$

with $\phi(\mathbf{X}_i)^T \Lambda_\tau \phi(\mathbf{X}_i) = \sum_{l=1}^L \tau_l \phi_l(\mathbf{X}_i)^T \phi_l(\mathbf{X}_i) = \sum_{l=1}^L \tau_l \mathbf{K}_i^{(l)} = \mathbf{K}_i^\tau$, where $\mathbf{K}_i^{(l)}, \mathbf{K}_i^\tau \in \mathbb{R}^{n_i \times n_i}$ are the base and combined kernel matrices over $\mathbf{x}_j \in \mathcal{N}_i$, respectively, i.e., $\mathbf{K}_i^{(l)} = [\mathcal{K}^{(l)}(\mathbf{x}_u, \mathbf{x}_v)]$ and $\mathbf{K}_i^\tau = [\mathcal{K}^\tau(\mathbf{x}_u, \mathbf{x}_v)]$, for $\mathbf{x}_u, \mathbf{x}_v \in \mathcal{N}_i$. For fixed τ constrained on the simplex, the convex combination of the positive semi-definite kernel matrices : $\mathbf{K}_i^\tau = \sum_{l=1}^L \tau_l \mathbf{K}_i^{(l)}$ is still a positive semi-definite kernel matrix. Therefore, the problem in (13) is an unconstrained concave quadratic program whose unique optimal solution can be obtained analytically:

$$\gamma_i^{c*} = 2\beta(\mathbf{I}_i + \beta \Pi_i \mathbf{K}_i^\tau \Pi_i)^{-1} \Pi_i \mathbf{y}_i^c. \quad (14)$$

Then altogether with (8), (12) and (14), the predicted indicator value at point \mathbf{x}_i for the c th ($c = 1, \dots, C$) cluster can be calculated by (6): $\hat{y}_{ic} = f_i^c(\phi(\mathbf{x}_i)) = \phi(\mathbf{x}_i)^T \mathbf{w}_i^c + b_i^c = \boldsymbol{\alpha}_i^T \mathbf{y}_i^c$, with

$$\boldsymbol{\alpha}_i^T = \beta(\mathbf{k}_i^\tau - \frac{1}{n_i} \mathbf{e}_i^T \mathbf{K}_i^\tau) \Pi_i \left[\mathbf{I}_i - (\beta^{-1} \mathbf{I} + \Pi_i \mathbf{K}_i^\tau \Pi_i)^{-1} \Pi_i \mathbf{K}_i^\tau \Pi_i \right] + \frac{1}{n_i} \mathbf{e}_i^T, \quad (15)$$

where $\mathbf{k}_i^\tau \in \mathbb{R}^{n_i}$ denotes the vector $[\mathcal{K}^\tau(\mathbf{x}_i, \mathbf{x}_j)]^T$ for $\mathbf{x}_j \in \mathcal{N}_i$.

To obtain \mathbf{Y} , we will first build the matrix \mathbf{T} by (3) with $\boldsymbol{\alpha}_i$ defined in (15), using the combined kernel $\mathcal{K}^\tau(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^L \tau_l \mathcal{K}^{(l)}(\mathbf{x}_i, \mathbf{x}_j)$. Then \mathbf{Y} is given by the first C eigenvectors of \mathbf{T} corresponding to the C smallest eigenvalues.

3.2 Update τ for a Given \mathbf{Y}

Subsequently, the L kernel combination coefficients $\{\tau_l\}_{l=1}^L$ will be recomputed based on the current estimation for \mathbf{Y} . We propose to estimate τ using the *projected gradient descent* method as in [11][12].

With fixed \mathbf{Y} and neighborhood determined at each point, an optimal τ is expected to minimize:

$$\mathcal{P}(\tau), \text{ s. t. } \sum_{l=1}^L \tau_l = 1, \tau_l \geq 0, \forall l, \quad (16)$$

where $\mathcal{P}(\boldsymbol{\tau}) = \min_{\mathbf{w}_i^c} \sum_{c=1}^C \sum_{i=1}^n \left[\beta \|\boldsymbol{\Pi}_i \mathbf{y}_{ic} - (\phi(\mathbf{X}_i) \boldsymbol{\Pi}_i)^T \mathbf{w}_i^c\|^2 + \mathbf{w}_i^{cT} \boldsymbol{\Lambda}_\tau^{-1} \mathbf{w}_i^c \right]$. In general, it can be solved by the projected gradient descent method through the update equation $\boldsymbol{\tau}^{(new)} = \boldsymbol{\tau}^{(old)} - \eta \nabla \mathcal{P}$, as given \mathbf{Y} and \mathcal{N}_i such that $\mathcal{P}(\boldsymbol{\tau}^{(new)}) \leq \mathcal{P}(\boldsymbol{\tau}^{(old)})$, where η is the step size, and $\nabla \mathcal{P}$ is the projected gradient. It is expected that the local regression model derived from $\boldsymbol{\tau}^{(new)}$ should be better than the one derived from $\boldsymbol{\tau}^{(old)}$. Nevertheless, since both \mathbf{Y} and \mathcal{N}_i depend on $\boldsymbol{\tau}$ as shown in Section 3.1, they need to be recomputed as in Section 3.1 once $\boldsymbol{\tau}$ is updated.

Then the key issue is to obtain the derivatives of $\mathcal{P}(\boldsymbol{\tau})$ in analytic forms. In order to do so, we resort to the dual of $\mathcal{P}(\boldsymbol{\tau})$ which has been investigated in Sub-section 3.1, and is rewritten below:

$$\mathcal{D}(\boldsymbol{\tau}) = \max_{\boldsymbol{\gamma}_i^c} \sum_{c=1}^C \sum_{i=1}^n -\frac{1}{4\beta} \boldsymbol{\gamma}_i^{cT} \boldsymbol{\gamma}_i^c - \frac{1}{4} \boldsymbol{\gamma}_i^{cT} \boldsymbol{\Pi}_i \mathbf{K}_i^\top \boldsymbol{\Pi}_i \boldsymbol{\gamma}_i^c + \boldsymbol{\gamma}_i^{cT} \boldsymbol{\Pi}_i \mathbf{y}_i^c. \quad (17)$$

Note (9) is convex with respect to \mathbf{w}_i^c . By the principle of strong duality, we have $\mathcal{P}(\boldsymbol{\tau}) = \mathcal{D}(\boldsymbol{\tau})$. Furthermore, as $\{\boldsymbol{\gamma}_i^{c*}\}$ in (14) maximizes \mathcal{D} , according to (13), $\mathcal{D}(\boldsymbol{\tau})$ is differentiable if $\{\boldsymbol{\gamma}_i^{c*}\}$'s are unique. Fortunately, this unicity is guaranteed by the unconstrained concave quadratic program in (13). Moreover, as proved in Lemma 2 of (14), $\mathcal{D}(\boldsymbol{\tau})$ can be differentiated with respect to $\boldsymbol{\tau}$ as if $\{\boldsymbol{\gamma}_i^{c*}\}$ did not depend on $\boldsymbol{\tau}$. Finally, we have:

$$\frac{\partial \mathcal{P}}{\partial \tau_l} = \frac{\partial \mathcal{D}}{\partial \tau_l} = -\frac{1}{4} \sum_{c=1}^C \sum_{i=1}^n \boldsymbol{\gamma}_i^{c*T} \boldsymbol{\Pi}_i \mathbf{K}_i^{(l)} \boldsymbol{\Pi}_i \boldsymbol{\gamma}_i^{c*} = -\frac{1}{4} \sum_{i=1}^n \text{trace}(\boldsymbol{\gamma}_i^{*T} \boldsymbol{\Pi}_i \mathbf{K}_i^{(l)} \boldsymbol{\Pi}_i \boldsymbol{\gamma}_i^*), \quad (18)$$

where $\boldsymbol{\gamma}_i^* = [\boldsymbol{\gamma}_i^{1*}, \dots, \boldsymbol{\gamma}_i^{C*}] \in \mathbb{R}^{n_i \times C}$.

Note the equality and non-negative constraints over the $\boldsymbol{\tau}$ have to be kept inviolate when updating $\boldsymbol{\tau}$ along the descent gradient direction. We use the same strategy as in (12) by first projecting the gradient to enforce the equality, and then ensuring that the descent direction does not lead to negative τ_l . That is, each element of the reduced gradient $\nabla \mathcal{P}$ is designed as follows:

$$(\nabla \mathcal{P})_l = \begin{cases} \frac{\partial \mathcal{P}}{\partial \tau_l} - \frac{\partial \mathcal{P}}{\partial \tau_m}, & \text{if } l \neq m \text{ and } \tau_l > 0; \\ \sum_{\mu \neq m, \tau_\mu > 0} \left(\frac{\partial \mathcal{P}}{\partial \tau_m} - \frac{\partial \mathcal{P}}{\partial \tau_\mu} \right), & \text{if } l = m; \\ 0, & \text{if } \tau_l = 0 \text{ and } \frac{\partial \mathcal{P}}{\partial \tau_l} - \frac{\partial \mathcal{P}}{\partial \tau_m} > 0, \end{cases} \quad (19)$$

where $m = \arg \max_l \tau_l$. When updating $\boldsymbol{\tau}$ by $\boldsymbol{\tau}^{(new)} = \boldsymbol{\tau}^{(old)} - \eta \nabla \mathcal{P}$, we first try η with the maximal admissible step size η_{max} which sets τ_ν to zero, where

$$\nu = \arg \min_{\{l | (\nabla \mathcal{P})_l > 0\}} \frac{\tau_l^{(old)}}{(\nabla \mathcal{P})_l}, \quad \eta_{max} = \frac{\tau_\nu}{(\nabla \mathcal{P})_\nu}. \quad (20)$$

If $\mathcal{D}(\boldsymbol{\tau}^{(trial)}) \leq \mathcal{D}(\boldsymbol{\tau}^{(old)})$, where $\boldsymbol{\tau}^{(trial)} = \boldsymbol{\tau}^{(old)} - \eta_{max} \nabla \mathcal{P}$, $\boldsymbol{\tau}$ gets updated; otherwise, a one-dimensional line search for $\eta \in [0, \eta_{max}]$ is applied. Algorithm 1 describes the steps to update $\boldsymbol{\tau}$.

Algorithm 1. Update kernel weight vector τ with the current \mathbf{Y} and \mathcal{N}_i

Compute the projected gradient $\nabla \mathcal{P}$ by (19);
 Compute the maximal admissible step size η_{max} by (20);
 $\tau^{(trial)} = \tau^{(old)} - \eta_{max} \nabla \mathcal{P};$
 Compute $\mathcal{D}(\tau^{(trial)})$ with $\{\gamma_i^*\}$ calculated from $\mathbf{K}^{\tau^{(trial)}} = \sum_{l=1}^L \tau_l^{(trial)} \mathbf{K}^{(l)}$;
if $\mathcal{D}(\tau^{(trial)}) \leq \mathcal{D}(\tau^{(old)})$ **then**
 | $\eta = \eta_{max};$
else
 | Perform line search for $\eta \in [0, \eta_{max}]$ along $\nabla \mathcal{P};$
end
 $\tau^{(new)} = \tau^{(old)} - \eta \nabla \mathcal{P};$

3.3 The Complete Algorithm

The complete local learning based clustering algorithm with multiple kernel learning (denoted as LLC-mkl) is presented in Algorithm 2. The loop stops when the relative variation of the trace value in (4) between two consecutive iterations is below a threshold (we set it at 10^{-4} in this paper), indicating the partitioning has almost been stabilized. After the convergence, \mathbf{Y} is discretized to obtain the final clustering result with the k-means as in [9].

Algorithm 2. Multiple kernel learning for local learning based clustering algorithm

input : L base kernel matrices $\mathbf{K}^{(l)}$'s, size of the neighborhood k , trade-off parameter β
output: \mathbf{Y}, τ

- 1 Initialize $\tau_l = \frac{1}{L}$, for $l = 1, \dots, L$;
- 2 **while** not converge **do**
- 3 Find k -mutual neighborhoods, using the metric defined in (5);
- 4 Construct the matrix \mathbf{T} by (3) with α_i given in (15), and then solve the problem (4) to obtain \mathbf{Y} ;
- 5 Update τ with the steps described in Algorithm 1
- 6 **end**

4 Experimental Results

Experiments on document clustering were conducted with LLC-mkl. The characteristics of the benchmark document datasets used in this experiment are summarized in Table II.

- **CSTR:** This is the dataset of the abstracts of technical reports published in the Department of Computer Science at a university between 1991 and 2002. The dataset contains 476 abstracts, which are divided into four research topics.
- **WebACE:** This dataset is from WebACE project, and it contains 2340 documents consisting of news articles from Reuters news service with 20 different topics in October 1997.

Table 1. Characteristics of the document datasets

Dataset	Number of Samples (n)	Number of Classes (C)
CSTR	476	4
WebACE	2340	20
tr11	414	9
tr31	927	7

- **tr11** and **tr31**: Both of the two datasets are from the CLUTO toolkit [15], they contain 414 and 927 articles categorized into 9 and 7 topics, respectively.

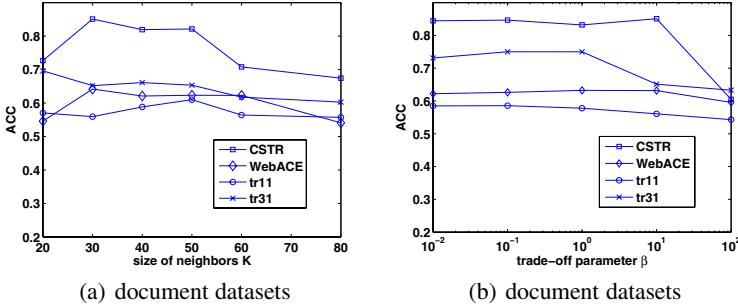
To pre-process the CSTR and WebACE datasets, we remove the stop words using a standard stop list, all HTML tags are skipped and all header fields except subject and organization of the posted articles are ignored. Then each document is represented by the term-frequency vector (Bag-of-Words). The datasets associated with the CLUTO toolkit have already been preprocessed. For all datasets, we used the top 1000 words by mutual information with class labels. For comparison, the counterpart unsupervised multiple kernel learning algorithm based on NMF [8] (denoted as NMF-mkl) was conducted. We also compared with the self-tuning spectral clustering [4] (denoted as Self-TunSpec), which tries to build a single best kernel for clustering. The algorithm in [7] is not compared because the optimization software in [7] cannot deal with the datasets that have too many samples and will cause memory overflow on the datasets used in this paper. Furthermore, we simply set the number of clusters equal to the number of classes in each dataset for all the algorithms without considering the selection of the optimal number of clusters, which is beyond the scope of this paper. We evaluated the performance with the clustering accuracy (ACC) index [11] for all algorithms. The sensitivity of the proposed LLC-mkl algorithm with respect to k and β will be presented at the end of this section.

We applied the LLC-mkl with altogether 10 pre-computed base kernels, i.e., 7 RBF kernels $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\delta^2)$, with $\delta = \text{const} * D$, where D is the maximum distance between samples, and const varies in the pre-specified range $\{0.01, 0.05, 0.1, 1, 10, 50, 100\}$, 2 polynomial kernels $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^d$ with degree $d = \{2, 4\}$, and a cosine kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j / (\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|)$. All the kernels have been normalized through: $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) / \sqrt{\mathcal{K}(\mathbf{x}_i, \mathbf{x}_i)\mathcal{K}(\mathbf{x}_j, \mathbf{x}_j)}$. Besides, we also implemented the case where each time a single candidate kernel $\mathcal{K}^{(l)} (l = 1, \dots, 10)$ was adopted in the LLC algorithm in which the local prediction is performed with kernel ridge regression. The best (denoted as LLC-bkernel) and the worst (denoted as LLC-wkernel) performance out of the 10 kernels were reported. NMF-mkl was applied on the same 10 base kernels. The adjacency matrix in SelfTunSpec [4] was built by its local scaling method [4] on the dataset. For NMF-mkl and SelfTunSpec, we only reported the best accuracy among extensive trials of their free parameters. For LLC-mkl, the mean and standard deviation of ACC with $k = 30, \beta = 10$ over 10 runs were reported. The results are summarized in Table 2.

From Table 2 we could first observe that there is a big gap between the best and the worst performance of LLC with different choices of kernel. On the tr11 and tr31

Table 2. Accuracies of various methods on the document datasets

Data Set	LLC-wkernel	LLC-bkernel	LLC-mkl	NMF-mkl	SelfTunSpec
CSTR	0.3487	0.7374	0.8508±0.0012	0.6387	0.5210
WebACE	0.2436	0.4885	0.6316±0.0215	0.4960	0.4880
tr11	0.4251	0.5966	0.5609±0.0166	0.5145	0.4106
tr31	0.5297	0.6721	0.6512±0.0007	0.5372	0.4412

**Fig. 1.** The parameter sensitivity studies of LLC-mkl algorithm. (a) varying the size of neighborhood with β being fixed at 10; (b) varying β with the size of neighborhood fixed at 30. The values on each line represent the average ACC over 10 independent runs.

datasets, the performance of LLC-mkl is close to that of the LLC with the best kernel, but obviously LLC-mkl is more sensible for practical application where we often do not know which kernel is the best *a priori*. On the CSTR and WebACE datasets, the LLC-mkl even outperforms the LLC with the best kernel. Namely, by combining multiple kernels and exploiting the complementary information contained in different kernels, the LLC-mkl indeed improves the robustness and accuracy of LLC. Compared to NMF-mkl which is derived globally, the LLC-mkl is consistently superior over it on these four datasets. A plausible reason is that the document datasets are very sparse, therefore the entries in the kernel matrix may resemble to each other from the global view or on a large scale. Thereby, finding the similar points locally may produce more reliable intermediate clustering result to guide the kernel learning. From Table 2, it can also be seen that the LLC-mkl and NMF-mkl both outperform the selfTunSpec which tries to construct a single “best” kernel in this experiment.

The effects of these two parameters, i.e., k and β , on the performance of LLC-mkl are presented in Figure 1. From Figure 1, it can be seen that the proposed LLC-mkl algorithm with $k = 30 \sim 50$ and $\beta \in [0.01, 10]$ could produce considerably accurate results and the performance does not vary much.

5 Conclusion

In this paper, a novel kernel learning approach has been proposed for the local learning based clustering, where a combination of kernels is jointly learned with the clustering.

It is addressed under a regularization framework by taking the relevance of each kernel into account. Experimental results have shown that the proposed kernel learning method is able to improve the robustness and accuracy of the basic local learning clustering. Furthermore, it generally outperforms the state-of-the-art counterparts, especially when the samples are less separable from a global view.

References

1. Wu, M., Schölkopf, B.: A Local Learning Approach for Clustering. In: NIPS, pp. 1529–1536 (2006)
2. Zeng, H., Cheung, Y.M.: Feature Selection for Local Learning based Clustering. In: PAKDD, pp. 414–425 (2009)
3. Schölkopf, B., Smola, A.J.: Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (2002)
4. Zelnik-Manor, L., Perona, P.: Self-tuning Spectral Clustering. In: NIPS, pp. 1601–1608 (2004)
5. Bach, F.R., Jordan, M.I.: Learning Spectral Clustering, with Application to Speech Separation. JMLR 7, 1963–2001 (2006)
6. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, M.I.E., Jordan, M.I.: Learning the Kernel Matrix with Semidefinite Programming. JMLR 5, 27–72 (2004)
7. Valizadegan, H., Jin, R.: Generalized Maximum Margin Clustering and Unsupervised Kernel Learning. In: NIPS, pp. 1417–1424 (2007)
8. Lange, T., Buhmann, J.: Fusion of Similarity Data in Clustering. In: NIPS, pp. 723–730 (2005)
9. Ng, A., Jordan, M., Weiss, Y.: On Spectral Clustering: Analysis and an Algorithm. In: NIPS, pp. 849–856 (2001)
10. Yu, S.X., Shi, J.: Multiclass Spectral Clustering. In: ICCV, pp. 313–319 (2003)
11. Calamai, P.H., Moré, J.J.: Projected Gradients Methods for Linearly Constrained Problems. Math. Prog., 93–116 (1987)
12. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: More Efficiency in Multiple Kernel Learning. In: ICML, pp. 775–782 (2007)
13. Bonnans, J.F., Shapiro, A.: Perturbation Analysis of Optimization Problems (2000)
14. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing Multiple Parameters for Support Vector Machines. Mach. Learn., 131–159 (2002)
15. Karypis, G.: CLUTO-A Clustering Toolkit (2002),
<http://www-users.cs.umn.edu/~karypis/cluto>

Projective Nonnegative Matrix Factorization with α -Divergence

Zhirong Yang and Erkki Oja

Department of Information and Computer Science*
Helsinki University of Technology
P.O. Box 5400, FI-02015, TKK, Espoo, Finland
{zhihong.yang,erkki.oja}@tkk.fi

Abstract. A new matrix factorization algorithm which combines two recently proposed nonnegative learning techniques is presented. Our new algorithm, α -PNMF, inherits the advantages of Projective Nonnegative Matrix Factorization (PNMF) for learning a highly orthogonal factor matrix. When the Kullback-Leibler (KL) divergence is generalized to α -divergence, it gives our method more flexibility in approximation. We provide multiplicative update rules for α -PNMF and present their convergence proof. The resulting algorithm is empirically verified to give a good solution by using a variety of real-world datasets. For feature extraction, α -PNMF is able to learn highly sparse and localized part-based representations of facial images. For clustering, the new method is also advantageous over Nonnegative Matrix Factorization with α -divergence and ordinary PNMF in terms of higher purity and smaller entropy.

1 Introduction

Nonnegative learning based on matrix factorization has received a lot of research attention recently. The first application of *Nonnegative Matrix Factorization* (NMF) [1] was in extracting sparse features of facial images, while recent research also reveals its usefulness in clustering.

However, the original NMF approximation is restricted to least square errors or the Kullback-Leibler divergence between the data matrix and its approximation. It has recently been pointed out that the divergence minimization can be generalized by using the α -divergence [2], which leads to a family of new algorithms [3]. The empirical study by Cichocki et al. shows that the generalized NMF can achieve better performance for various applications by using proper α values.

Projective Nonnegative Matrix Factorization (PNMF) [4] is another variant of NMF. It identifies a nonnegative subspace by integrating the nonnegativity to the PCA objective. PNMF has proven to outperform NMF in feature extraction, where PNMF is able to generate sparser patterns which are more localized and

* Supported by the Academy of Finland in the project *Finnish Centre of Excellence in Adaptive Informatics Research*.

non-overlapping [4]. Clustering results of text data also demonstrate that PNMF is advantageous as it provides better approximation to the binary-valued multi-cluster indicators than NMF.

To achieve both merits of the above methods, we extend the PNMF by using α -divergence instead of KL-divergence as the error measure. We derive the multiplicative update rules for the new learning objective. The convergence of the iterative updates is proven using the Lagrangian approach. Experiments are conducted, in which the new algorithm outperforms α -NMF for extracting sparse and localized part-based representations of facial images. Our method can also achieve better clustering results than α -NMF and ordinary PNMF for a variety of datasets.

2 Related Work

2.1 Nonnegative Matrix Factorization

Given a nonnegative data matrix $\mathbf{X} \in \mathbb{R}_+^{m \times N}$, *Nonnegative Matrix Factorization* (NMF) seeks a decomposition of \mathbf{X} that is of the form:

$$\mathbf{X} \approx \mathbf{WH}, \quad (1)$$

where $\mathbf{W} \in \mathbb{R}_+^{m \times r}$ and $\mathbf{H} \in \mathbb{R}_+^{r \times N}$ with the rank $r \ll \min(m, N)$.

Denote by $\hat{\mathbf{X}} = \mathbf{WH}$ the approximating matrix. The approximation can be achieved by minimizing two widely used measures: (1) Least Square criterion $\varepsilon = \sum_{i,j} (X_{ij} - \hat{X}_{ij})^2$ and (2) *Kullback-Leibler divergence* (KL-divergence)

$$D_{\text{KL}}(\mathbf{X} || \hat{\mathbf{X}}) = \sum_{i,j} \left(X_{ij} \log \frac{X_{ij}}{\hat{X}_{ij}} - X_{ij} + \hat{X}_{ij} \right). \quad (2)$$

In this paper we focus on the second approximation criterion, which leads to the multiplicative updating rules of the form

$$H_{kj}^{\text{new}} = H_{kj} \frac{(\mathbf{W}^T \mathbf{Z})_{kj}}{\sum_i W_{ik}}, \quad W_{ik}^{\text{new}} = W_{ik} \frac{(\mathbf{Z} \mathbf{H}^T)_{ik}}{\sum_j H_{kj}}, \quad (3)$$

where we use $Z_{ij} = X_{ij}/\hat{X}_{ij}$ for notational brevity.

2.2 Nonnegative Matrix Factorization with α -Divergence

The α -divergence [2] is a parametric family of divergence functionals, including several well-known divergence measures as special cases. NMF equipped with the following α -divergence as the approximation measure is called α -NMF [3].

$$D_\alpha(\mathbf{X} || \hat{\mathbf{X}}) = \frac{1}{\alpha(1-\alpha)} \sum_{i=1}^m \sum_{j=1}^N \left(\alpha X_{ij} + (1-\alpha)\hat{X}_{ij} - X_{ij}^\alpha \hat{X}_{ij}^{1-\alpha} \right) \quad (4)$$

The corresponding multiplicative update rules are given by the following, where we define $\tilde{Z}_{ij} = Z_{ij}^\alpha$:

$$H_{kj}^{\text{new}} = H_{kj} \left[\frac{(\mathbf{W}^T \tilde{\mathbf{Z}})_{kj}}{\sum_i W_{ik}} \right]^{\frac{1}{\alpha}}, \quad W_{ik}^{\text{new}} = W_{ik} \left[\frac{(\tilde{\mathbf{Z}} \mathbf{H}^T)_{ik}}{\sum_j H_{kj}} \right]^{\frac{1}{\alpha}}. \quad (5)$$

α -NMF reduces to the conventional NMF with KL-divergence when $\alpha \rightarrow 1$. Another choice of α characterizes a different learning principle, in the sense that the model distribution is more inclusive ($\alpha \rightarrow \infty$) or more exclusive ($\alpha \rightarrow -\infty$). Such flexibility enables α -NMF to outperform NMF with α properly selected.

2.3 Projective Nonnegative Matrix Factorization

Replacing $\mathbf{H} = \mathbf{W}^T \mathbf{X}$ in (1), we get the *Projective Nonnegative Matrix Factorization* (PNMF) approximation scheme [4]

$$\mathbf{X} \approx \mathbf{W} \mathbf{W}^T \mathbf{X}. \quad (6)$$

Again, denote $\hat{\mathbf{X}} = \mathbf{W} \mathbf{W}^T \mathbf{X}$ the approximating matrix and $Z_{ij} = X_{ij}/\hat{X}_{ij}$. The PNMF multiplicative update rule for KL-divergence is given by [4]

$$W_{ik}^{\text{new}} = W_{ik} \frac{(\mathbf{Z} \mathbf{X}^T \mathbf{W} + \mathbf{X} \mathbf{Z}^T \mathbf{W})_{ik}}{\sum_j (\mathbf{W}^T \mathbf{X})_{kj} + (\sum_j X_{ij}) (\sum_b W_{bk})}. \quad (7)$$

The name PNMF comes from another derivation of the approximation scheme (6) where a projection matrix \mathbf{P} in $\mathbf{X} \approx \mathbf{P} \mathbf{X}$ is factorized into $\mathbf{W} \mathbf{W}^T$. This interpretation connects PNMF with the classical *Principal Component Analysis* subspace method except for the nonnegativity constraint [4]. Compared with NMF, PNMF is able to learn a much sparser matrix \mathbf{W} . This property is especially desired for extracting part-based representations of data samples or finding cluster indicators.

3 PNMF with α -Divergence

In this section we combine the flexibility of α -NMF and the sparsity of PNMF into a single algorithm. We called the resulting method α -PNMF which stands for Projective Nonnegative Matrix Factorization with α -divergence.

3.1 Multiplicative Update Rule

α -PNMF solves the following optimization problem:

$$\underset{\mathbf{W} \geq \mathbf{0}}{\text{minimize}} \quad \mathcal{J}(\mathbf{W}) = D_\alpha(\mathbf{X} || \mathbf{W} \mathbf{W}^T \mathbf{X}). \quad (8)$$

The derivative of the objective with respect to \mathbf{W} is

$$\begin{aligned} \frac{\partial \mathcal{J}(\mathbf{W})}{\partial W_{ik}} = & \frac{1}{\alpha} \left[- \left(\tilde{\mathbf{Z}}\mathbf{X}^T\mathbf{W} + \mathbf{X}\tilde{\mathbf{Z}}^T\mathbf{W} \right)_{ik} \right. \\ & \left. + \sum_j (\mathbf{W}^T\mathbf{X})_{kj} + \left(\sum_j X_{ij} \right) \left(\sum_b W_{bk} \right) \right] \end{aligned} \quad (9)$$

Denote Λ_{ik} the Lagrangian multipliers associated with the constraint $W_{ik} \geq 0$. The Karush-Kuhn-Tucker (KKT) conditions require

$$\frac{\partial \mathcal{J}(\mathbf{W})}{\partial W_{ik}} = \Lambda_{ik} \quad (10)$$

and $\Lambda_{ik}W_{ik} = 0$ which indicates $\Lambda_{ik}W_{ik}^\alpha = 0$. Multiplying both sides of (10) by W_{ik}^α leads to $\frac{\partial \mathcal{J}(\mathbf{W})}{\partial W_{ik}}W_{ik}^\alpha = 0$. This suggests a multiplicative update rule by writing $\tilde{Z}_{ij} = Z_{ij}^\alpha$:

$$W'_{ik} = W_{ik} \left[\frac{\left(\tilde{\mathbf{Z}}\mathbf{X}^T\mathbf{W} + \mathbf{X}\tilde{\mathbf{Z}}^T\mathbf{W} \right)_{ik}}{\sum_j (\mathbf{W}^T\mathbf{X})_{kj} + \left(\sum_j X_{ij} \right) \left(\sum_a W_{ak} \right)} \right]^{\frac{1}{\alpha}}. \quad (11)$$

3.2 Convergence Proof

The convergence of NMF and most of its variants, including α -NMF, to a local minimum of the cost function is analyzed by using an auxiliary function [3]. It is however difficult to directly construct such a function for α -PNMF because of the auto-association induced by \mathbf{W} and its transpose. Here we overcome this problem by applying the Lagrangian technique to decouple the auto-association.

With the constraint $\mathbf{H} = \mathbf{W}^T\mathbf{X}$, one can write the Lagrangian objective function as

$$\mathcal{L}(\mathbf{W}, \mathbf{H}) = D_\alpha(\mathbf{X} || \mathbf{WH}) + \text{Tr} \left(\Psi^T (\mathbf{H} - \mathbf{W}^T\mathbf{X}) \right), \quad (12)$$

by introducing multipliers Ψ_{ik} . Following [3], we apply Jensen's inequality using the convex function $f(z) = [\alpha + (1 - \alpha)z - z^{1-\alpha}] / (\alpha(1 - \alpha))$ for $\alpha \geq 0$, which leads to

$$f \left(\sum_k W_{ik} H_{kj} \right) \leq \sum_k \zeta_{ijk} f \left(\frac{W_{ik} H_{kj}}{X_{ij} \zeta_{ijk}} \right), \quad (13)$$

with $\zeta_{ijk} = \frac{W_{ik} H_{kj}}{\sum_l W_{il} H_{lj}}$. After some manipulation, one can find that $\mathcal{L}(\mathbf{W}, \mathbf{H})$ is upper-bounded by the auxiliary function

$$\begin{aligned}
G(\mathbf{W}', \mathbf{W}) = & \frac{1}{\alpha} \sum_{i,j,k} X_{ij} \zeta_{ijk} \left[\alpha + (1-\alpha) \frac{W'_{ik} H_{kj}}{X_{ij} \zeta_{ijk}} - \left(\frac{W'_{ik} H_{kj}}{X_{ij} \zeta_{ijk}} \right)^{1-\alpha} \right] \\
& + \text{Tr} \left(\Psi^T (\mathbf{H} - \mathbf{W}^T \mathbf{X}) \right) \\
& + \sum_{ik} \left(\mathbf{X} \tilde{\mathbf{Z}}^T \mathbf{W} \right)_{ik} \left(W'_{ik} - W_{ik} - W_{ik} \log \frac{W'_{ik}}{W_{ik}} \right). \quad (14)
\end{aligned}$$

The last line of eq. (14) is a tight upper-bound of zero. To see this, one can insert $y = W'_{ik}/W_{ik}$ into the inequality $y \geq 1 + \log y$ for $y \geq 0$, where the equality holds if and only if $y = 1$. This additional bounding aims to add the same term $(\mathbf{X} \tilde{\mathbf{Z}}^T \mathbf{W})_{ik}$ to both numerator and denominator of the resulting multiplicative update rule and thus maintains the nonnegativity of \mathbf{W} .

Setting $\partial G / \partial \mathbf{W}' = 0$, we get

$$\left(\frac{W'_{ik}}{W_{ik}} \right)^\alpha = \frac{\left(\tilde{\mathbf{Z}} \mathbf{H}^T \right)_{ik} + \left(\mathbf{X} \tilde{\mathbf{Z}}^T \mathbf{W} \right)_{ik}}{\sum_j H_{kj} - \alpha \left(\mathbf{X} \Psi^T \right)_{ik} + \left(\mathbf{X} \tilde{\mathbf{Z}}^T \mathbf{W} \right)_{ik}}. \quad (15)$$

Next we solve Ψ by using the KKT conditions. From

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{H})}{\partial H_{kj}} = \frac{1}{\alpha} \left(\sum_i W_{ik} - \left(\mathbf{W}^T \tilde{\mathbf{Z}} \right)_{kj} \right) + \Psi_{kj} = 0 \quad (16)$$

we get

$$\alpha \left(\mathbf{X} \Psi^T \right)_{ik} = \left(\mathbf{X} \tilde{\mathbf{Z}}^T \mathbf{W} \right)_{ik} - \left(\sum_j X_{ij} \right) \left(\sum_b W_{bk} \right) \quad (17)$$

Inserting (17) and $\mathbf{H} = \mathbf{W}^T \mathbf{X}$ into (15), one obtains the multiplicative update rule (11). This concludes our proof of the following result:

Theorem 1. $D_\alpha(\mathbf{X} || \mathbf{W} \mathbf{W}^T \mathbf{X})$ is non-increasing under the multiplicative updates using (11).

3.3 Stabilization

The multiplicative updates can start from any initial guess of \mathbf{W} . However, we find some initial values may lead to a very zigzag convergence path. The overall scaling of \mathbf{W} greatly fluctuates between odd and even iterations.

We propose to overcome this problem by introducing one more parameter ρ . The modified objective becomes to minimize $\tilde{\mathcal{J}}(\rho, \mathbf{W}) = D_\alpha(\mathbf{X} || \rho \mathbf{W} \mathbf{W}^T \mathbf{X})$. Fixing \mathbf{W} , the global optimal ρ^* can be solved by setting the derivative of $\tilde{\mathcal{J}}(\rho, \mathbf{W})$ with respect to ρ to zero, which results in

$$\rho^* = \left(\frac{\sum_{ij} \hat{X}_{ij} \tilde{Z}_{ij}}{\sum_{ij} \hat{X}_{ij}} \right)^{\frac{1}{\alpha}} \quad (18)$$

Next, fixing ρ the optimal \mathbf{W} given its current estimate can be found by inserting ρ^* in the denominator of (II). Equivalently, one can apply the original multiplicative update rule and then compute

$$W_{ik}^{\text{new}} = W_{ik}' \left(\frac{\sum_{ij} \hat{X}_{ij} \tilde{Z}_{ij}}{\sum_{ij} \hat{X}_{ij}} \right)^{\frac{1}{2\alpha}} \quad (19)$$

with re-calculated $\hat{\mathbf{X}}$ and $\tilde{\mathbf{Z}}$.

If $\mathbf{W}\mathbf{W}^T\mathbf{X}$ approximates \mathbf{X} well, all the \tilde{Z}_{ij} approach one and so does ρ^* . The modified objective is thus equivalent to the original one. Therefore ρ serves as an intermediate variable that stabilizes and speeds up the algorithm especially in early iterations.

4 Experiments

Suppose the nonnegative matrix $\mathbf{X} \in \mathbb{R}_+^{m \times N}$ is composed of N data samples $\mathbf{x}_j \in \mathbb{R}_+^m$, $j = 1, \dots, N$. Basically, α -PNMF can be applied on this matrix in two different ways. One employs the approximation scheme $\mathbf{X} \approx \mathbf{W}\mathbf{W}^T\mathbf{X}$ and performs *feature extraction* by projecting each sample into a nonnegative subspace. The other approach approximates the transposed matrix \mathbf{X}^T by $\mathbf{W}\mathbf{W}^T\mathbf{X}^T$ where $\mathbf{W} \in \mathbb{R}_+^{N \times r}$. The latter approach can be used for *clustering* where the elements of \mathbf{W} indicate the membership of each sample to the r clusters.

4.1 Feature Extraction

We have used the FERET database of facial images [5] as the training data set. After face segmentation, 2,409 frontal images (poses “fa” and “fb”) of 867 subjects were stored in the database for the experiments. All face boxes were normalized to the size of 32×32 and then reshaped to a 1024-dimensional vector by column-wise concatenation. Thus we obtained a 1024×2409 nonnegative data matrix, whose elements are re-scaled into the region $[0,1]$ by dividing with their maximum. For good visualization, we empirically set $r = 25$ in the feature extraction experiments.

After training, the basis vectors are stored in the columns of \mathbf{W} in α -NMF and α -PNMF. The basis vectors have same dimensionality with the image samples and thus can be visualized as *basis images*. In order to encode the features of different facial parts, it is expected to find some localized and non-overlapping patterns in the basis images. The resulting basis images using $\alpha = 0.5$ (Hellinger divergence), $\alpha = 1$ (KL-divergence) and $\alpha = 2$ (χ^2 -divergence) are shown in Figure I. Both methods can identify some facial parts such as eyebrows and lips. In comparison, α -PNMF is able to generate much sparser basis images with more part-based visual patterns.

Notice that two non-negative vectors are orthogonal if and only if they do not have the same non-zero dimensions. Therefore we can quantify the sparsity of the basis vectors by measuring their orthogonalities with the τ measurement:

$$\tau = 1 - \frac{\|\mathbf{R} - \mathbf{I}\|_F}{(r(r-1))}, \quad (20)$$

where $\|\cdot\|_F$ is the Frobenius matrix norm and R_{st} the normalized inner product between two basis vectors \mathbf{w}_s and \mathbf{w}_t :

$$R_{st} = \frac{\mathbf{w}_s^T \mathbf{w}_t}{\|\mathbf{w}_s\| \|\mathbf{w}_t\|}. \quad (21)$$

Larger τ 's indicate higher orthogonality and τ reaches 1 when the columns of \mathbf{W} are completely orthogonal. The orthogonalities using the two compared methods are displayed under the respective basis image plots in Figure 11. All τ values in the right are larger than their left counterparts, which confirms that α -PNMF is able to extract a sparser transformation matrix \mathbf{W} . It is worth to notice that α -PNMF achieves the high sparseness without the explicit orthogonality constraint compared with some other exiting methods such as [6].

4.2 Clustering

We have used a variety of datasets, most of which are frequently used in machine learning and information retrieval research. Table 11 summarizes the characteristics of the datasets. The descriptions of these datasets are as follows:

- *Iris*, *Ecoli5*, *WDBC*, and *Pima*, which are taken from the UCI data repository with respective datasets Iris, Ecoli, Breast Cancer Wisconsin (Prognostic), and Pima Indians Diabetes. The *Ecoli5* dataset contains only samples of the five largest classes in the original Ecoli database.
- *AMLALL* gene expression database [7]. This dataset contains acute lymphoblastic leukemia (ALL) that has B and T cell subtypes, and acute myelogenous leukemia (AML) that occurs more commonly in adults than in children. The data matrix consists of 38 bone marrow samples (19 ALL-B, 8 ALL-T and 11 AML) with 5000 genes as their dimensions.
- *ORL* database of facial images [8]. There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions and facial details. In our experiments, we down-sampled the images to size 46×56 and rescaled the gray-scale values to $[0, 1]$.

The number of clusters r is generally set to the number of classes. This work focuses on cases where $r > 2$, as there exist closed form approximations for the two-way clustering solution (see e.g. [9]). We thus set r equal to five times of the number of classes for *WDBC* and *Pima*.

Suppose there is ground truth data that labels the samples by one of q classes. We have used the *purity* and *entropy* measures to quantify the performance of the compared clustering algorithms:

$$\text{purity} = \frac{1}{N} \sum_{k=1}^r \max_{1 \leq l \leq q} n_k^l, \quad \text{entropy} = -\frac{1}{n \log_2 q} \sum_{k=1}^r \sum_{l=1}^q n_k^l \log_2 \frac{n_k^l}{n_k},$$

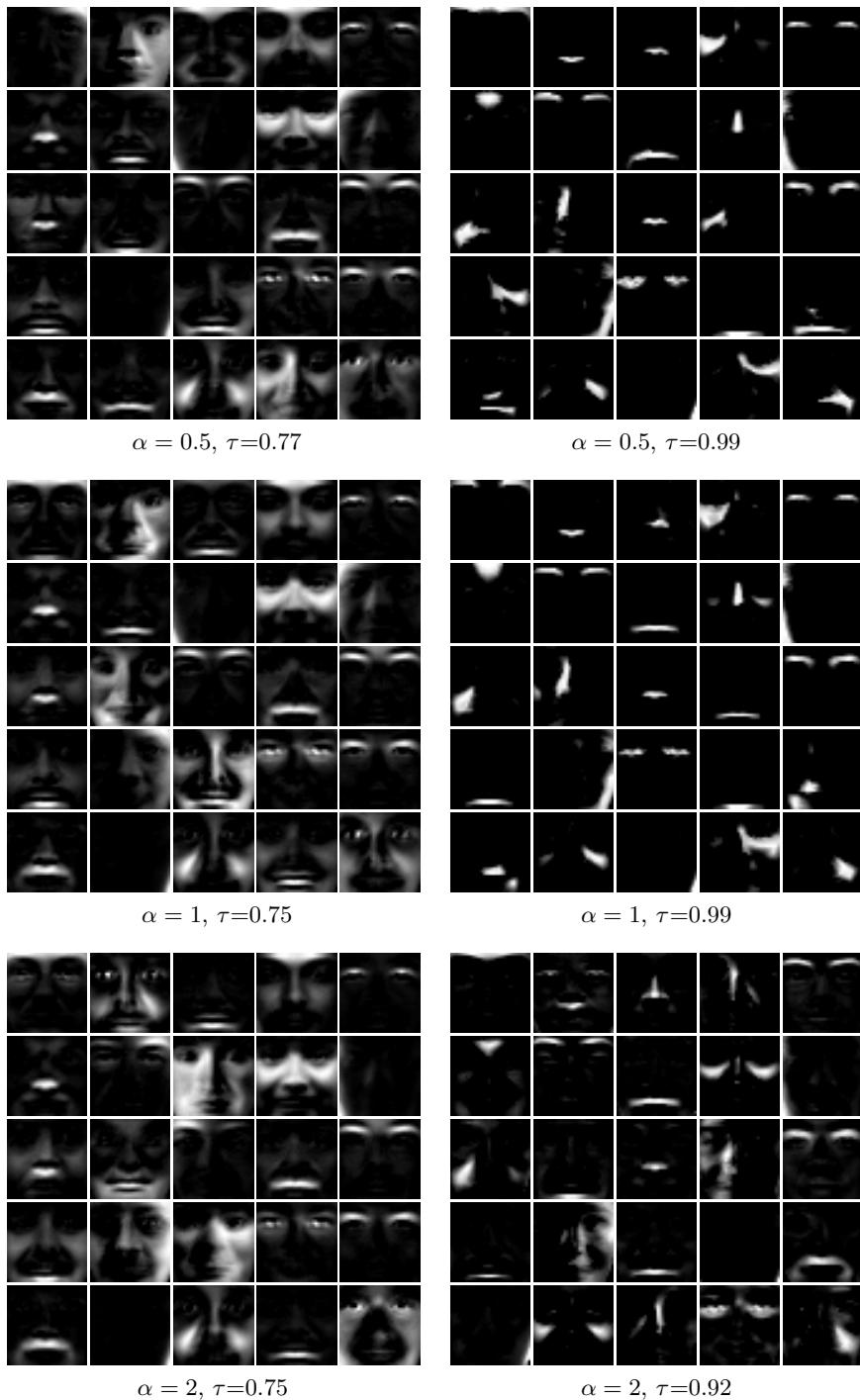


Fig. 1. The basis images of (left) α -NMF and (right) α -PNMF

Table 1. Dataset descriptions

datasets	#samples	#dimensions	#classes	r
Iris	150	4	3	3
Ecoli5	327	7	5	5
WDBC	569	30	2	10
Pima	768	8	2	10
AMLALL	38	5000	3	3
ORL	400	2576	40	40

Table 2. Clustering (a) purities and (b) entropies using α -NMF, PNMF and α -PNMF. The best result for each dataset is highlighted with boldface font.

(a)

datasets	α -NMF			PNMF	α -PNMF		
	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$	-	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$
Iris	0.83	0.85	0.84	0.95	0.95	0.95	0.97
Ecoli5	0.62	0.65	0.67	0.72	0.72	0.72	0.73
WDBC	0.70	0.70	0.72	0.87	0.86	0.87	0.88
Pima	0.65	0.65	0.65	0.65	0.67	0.65	0.67
AMLALL	0.95	0.92	0.92	0.95	0.97	0.95	0.92
ORL	0.47	0.47	0.47	0.75	0.76	0.75	0.80

(b)

datasets	α -NMF			PNMF	α -PNMF		
	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$	-	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$
Iris	0.34	0.33	0.33	0.15	0.15	0.15	0.12
Ecoli5	0.46	0.58	0.50	0.40	0.40	0.40	0.40
WDBC	0.39	0.38	0.37	0.16	0.17	0.16	0.14
Pima	0.92	0.90	0.90	0.91	0.90	0.91	0.89
AMLALL	0.16	0.21	0.21	0.16	0.08	0.16	0.21
ORL	0.35	0.34	0.35	0.14	0.14	0.14	0.12

where n_k^l is the number of samples in the cluster k that belong to original class l and $n_k = \sum_l n_k^l$. A larger purity value and a smaller entropy indicate better clustering performance.

The resulting purities and entropies are shown in Table 2, respectively. α -PNMF performs the best for all selected datasets. Recall that when $\alpha = 1$ the proposed method reduces to PNMF and thus returns results identical to the latter. Nevertheless, α -PNMF can outperform PNMF by adjusting the α value. When $\alpha = 0.5$, the new method achieves the highest purity and lowest entropy for the gene expression dataset *AMLALL*. For the other five datasets, one can set $\alpha = 2$ and obtain the best clustering result using α -PNMF. In addition, one can see that Nonnegative Matrix Factorization with α -divergence works poorly in our clustering experiments, much worse than the other methods. This

is probably because α -NMF has to estimate many more parameters than those using projective factorization. α -NMF is therefore prone to falling into bad local optima.

5 Conclusions

We have presented a new variant of NMF by introducing the α -divergence into the PNMF algorithm. Our α -PNMF algorithm theoretically converges to a local minimum. The resulting factor matrix is of high sparsity or orthogonality, which is desired for part-based feature extraction and multi-way clustering. Experimental results with various datasets indicate that the proposed algorithm can be considered as a promising replacement for α -NMF and PNMF for feature extraction and clustering.

References

1. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 788–791 (1999)
2. Amari, S.: Differential-geometrical methods in statistics. *Lecture Notes in Statistics*, vol. 28. Springer, New York (1985)
3. Cichocki, A., Lee, H., Kim, Y.D., Choi, S.: Non-negative matrix factorization with α -divergence. *Pattern Recognition Letters* 29, 1433–1440 (2008)
4. Yuan, Z., Oja, E.: Projective nonnegative matrix factorization for image compression and feature extraction. In: Kalviainen, H., Parkkinen, J., Kaarna, A. (eds.) *SCIA 2005. LNCS*, vol. 3540, pp. 333–342. Springer, Heidelberg (2005)
5. Phillips, P.J., Moon, H., Rizvi, S.A., Rauss, P.J.: The FERET evaluation methodology for face recognition algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence* 22, 1090–1104 (2000)
6. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 126–135 (2006)
7. Brunet, J.P., Tamayo, P., Golub, T.R., Mesirov, J.P.: Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences* 101(12), 4164–4169 (2004)
8. Samaria, F., Harter, A.: Parameterisation of a stochastic model for human face identification. In: Proceedings of 2nd IEEE Workshop on Applications of Computer Vision, Sarasota FL, December 1994, pp. 138–142 (1994)
9. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8), 888–905 (2000)

Active Generation of Training Examples in Meta-Regression

Ricardo B.C. Prudêncio and Teresa B. Ludermir

Center of Informatics, Federal University of Pernambuco,
CEP 50732-970 - Recife (PE), Brazil
`{rbcp,tbl}@cin.ufpe.br`

Abstract. Meta-Learning predicts the performance of learning algorithms based on features of the learning problems. Meta-Learning acquires knowledge from a set of meta-examples, which store the experience obtained from applying the algorithms to problems in the past. A limitation of Meta-Learning is related to the generation of meta-examples. In order to construct a meta-example, it is necessary to empirically evaluate the algorithms on a given problem. Hence, the generation of a set of meta-examples may be costly depending on the context. In order to minimize this limitation, the use of Active Learning is proposed to reduce the number of required meta-examples. In this paper, we evaluate this proposal on a promising Meta-Learning approach, called Meta-Regression. Experiments were performed in a case study to predict the performance of learning algorithms for MLP networks. A significant performance gain was observed in the case study when Active Learning was used to support the generation of meta-examples.

1 Introduction

Meta-Learning is a framework developed in the field of supervised machine learning with the aim of relating features of the learning problems to the performance of the learning algorithms [1]. The knowledge in Meta-Learning is acquired from a set of *meta-examples*, in which each meta-example stores the experience obtained from the application of a set of candidate algorithms in a particular learning problem. Meta-Learning automatically captures the expertise gained on different problems, which will be used to predict learning performance [2].

Different Meta-Learning approaches have been proposed [1]. An approach that revealed promising results in the literature is the Meta-Regression [3][4][5]. In this approach, each meta-example is related to a problem and stores: (1) features of the problem (e.g., number of training examples and attributes); and (2) the performance information (e.g., classification accuracy) estimated from the empirical evaluation of a set of candidate algorithms on the problem. In Meta-Regression, a regression algorithm is used to predict the performance information based on the features of the problems.

Generating a set of training examples for Meta-Regression may be a costly process, since in order to produce a single meta-example, it is necessary to perform an empirical evaluation of the candidate algorithms on a problem. Hence,

the cost of generating a whole set of meta-examples may be high, depending, for instance, on the number and complexity of the candidate algorithms, the methodology of empirical evaluation and the amount of available problems.

In order to minimize the above difficulty, the use of Active Learning [6] is proposed to support the generation of meta-examples for Meta-Regression. Active Learning is mainly focused on domains of application in which it is hard to produce training examples, which is the case of Meta-Regression. In our context, Active Learning techniques are used to select the most relevant problems for generating meta-examples, consequently reducing the effort in performing empirical evaluations on the candidate algorithms.

In our work, a prototype was implemented in order to evaluate the proposed approach. In this prototype, the Linear Regression (LR) algorithm was used as meta-regressor to predict the performance difference between two algorithms for training Multi-Layer Perceptron (MLP) neural networks [7]. An active technique based on uncertainty of prediction was applied in the selection of problems for meta-example generation. Experiments were performed on a set of 50 problems, comparing the active technique to a random (passive) strategy for selecting problems. The performed experiments revealed a performance gain in the meta-regressor when the active learning technique was considered.

Section 2 brings a brief presentation of Meta-Learning, followed by section 3 which describes the proposed solution and the implemented prototype. Section 4 presents the performed experiments and obtained results. Finally, section 5 concludes the paper by presenting some final considerations and future work.

2 Meta-Learning

Meta-Learning is focused in our work as the automatic process of acquiring knowledge that relates the performance of learning algorithms to the features of the learning problems [1]. In this context, each *meta-example* is related to a learning problem and stores: (1) the features describing the problem, called *meta-features*; and (2) information about the performance of one or more algorithms when applied to the problem. The *meta-learner* is a learning system that receives as input a set of such meta-examples and then acquires knowledge used to predict the algorithms performance for new problems being solved.

The meta-features are, in general, statistics describing the training dataset of the problem, such as number of training examples, number of attributes, correlation between attributes, class entropy, among others [8,9]. In a strict formulation of Meta-Learning, each meta-example stores, as performance information, a class label which indicates the best algorithm for the problem, among a set of candidates [10]. In this case, the class label for each meta-example is defined by performing a cross-validation experiment using the available dataset. The meta-learner is simply a classifier which predicts the best algorithm based on the meta-features of the problem.

Although the strict Meta-Learning approach (as described above) has been applied by different authors (such as [11,12,13]), certain information loss may

be introduced in the definition of the class labels associated to meta-examples. For instance, the performance of two algorithms may be very similar, and this information will be lost by merely recording the best algorithm as class label [5].

In order to overcome the above difficulty, the Meta-Regression approach [3][5][4] tries to directly predict a numerical performance measure (e.g., accuracy, error difference, rank,...) of the candidate algorithms. In this case, the meta-examples store as performance information the numerical performance obtained in previous problems. The meta-learner, in turn, is a regression model that may be used either to select the best candidate algorithm based on the highest predicted performance or to provide a ranking of algorithms based on the order of predicted performances.

In [3], the authors evaluated different algorithms as meta-regressors, including linear regression models, piecewise linear models, decision trees and instance-based regression. In [4], the authors used linear regression models to predict the accuracy of 8 classification algorithms, and the experiments revealed good results. In [5], the authors performed comparative experiments with both the strict Meta-Learning and Meta-Regression approaches, and observed that the latter one performed better when used to support algorithm selection.

3 Active Generation of Meta-examples

As said in section 2, Meta-Learning accumulates the experience stored in meta-examples which are derived from the empirical evaluation of the candidate algorithms on problems solved in the past. The generation of a set of meta-examples may be a costly process depending, for instance, on the complexity of the candidate algorithms, the cost of the methodology used for empirical evaluation, and the number and size of the available problems.

In order to minimize the above difficulty, Active Learning [6] was proposed in [4] in order to select the most relevant problems for meta-example generation, aiming to reduce the effort in performing empirical evaluations with the candidate algorithms. Active Learning is a paradigm in which the learning algorithm has some control over the inputs on which it trains [6]. Active Learning is ideal for domains in which the acquisition of labeled examples is a costly process. The use of Active Learning techniques in our context aims to reduce the number of meta-examples, and consequently the number of empirical evaluations of the algorithms, at same time maintaining the performance of the meta-learner.

Figure 1 represents the proposed approach. Initially, the meta-features are computed for each available problem, in order to generate a set of *unlabeled* meta-examples. Each unlabeled meta-example stores the description of a problem, but the performance information of the candidate algorithms is not known yet. In order to generate *labeled* meta-examples, the Active Learning module selects those unlabeled meta-examples considered the most relevant for the Meta-Learning task. The selection of unlabeled meta-examples is performed based on a pre-defined Active Learning method implemented in the module. Given the selected unlabeled meta-example, the candidate algorithms are then empirically evaluated on the related problem, in order to collect the performance

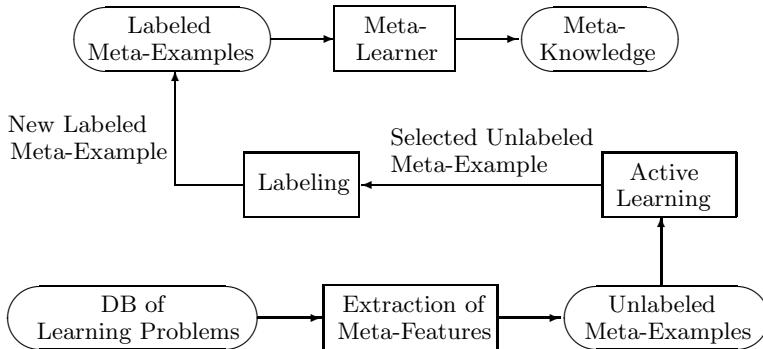


Fig. 1. Active generation of meta-examples

information. Each new labeled meta-example (composed of meta-features and performance information) is then stored in the training set of the Meta-Learner module. This module in turn will use this training set to acquire knowledge relating meta-features to the performance of the candidate algorithms.

The viability of this proposal was verified in [14] through a number of experiments in which an uncertainty sampling method improved the performance of a strict meta-learner. In this paper, we extend our previous work, by evaluating the use of Active Learning for Meta-Regression. As said, Meta-Regression is more flexible when compared to the strict Meta-Learning, and it has been a topic of interest for different authors, which motivated the current work. We highlight that the active techniques evaluated in our previous work [14] are not suitable to Meta-Regression, and hence, new techniques were investigated in the current research.

In order to evaluate the *Active Meta-Regression*, a prototype was implemented and applied to a meta-learning task which corresponds to predict the difference in performance between the Backpropagation (BP) [7] and the Levenberg-Marquardt (LM) [15] algorithms used to train MLP neural networks. The prediction of this performance information supports the choice between BP and LM to new problems at hand. In this task, each meta-example stored: (1) the value of 10 descriptive meta-features of a regression problem; and (2) the error difference between BP and LM observed in the problem. In this prototype, the Linear Regression (LR) algorithm was used as meta-regressor to predict the performance information based on the meta-features. An active method based on uncertainty of prediction was used to select meta-examples for the LR algorithm.

In the next subsections, we present details about the implemented prototype such as the definition of the meta-examples (section 3.1), the description of the meta-regressor (section 3.2) and the description of the Active Learning method (section 3.3).

3.1 Meta-examples

The set of meta-examples is generated from the application of the candidate algorithms on a set of learning problems. Formally, let $E = \{e_1, \dots, e_n\}$ be the set of n problems used to generate a set of n meta-examples $ME = \{me_1, \dots, me_n\}$. Each meta-example is related to a single problem and stores the values of p features X_1, \dots, X_p for the problem and the value of a target attribute Y . In this way, each meta-example $me_i \in ME$ is represented as the pair (x_i, y_i) storing: (1) the description x_i of the problem e_i , where $x_i = (x_i^1, \dots, x_i^p)$ and $x_i^j = X_j(e_i)$; and (2) the performance information y_i estimated for e_i , where $y_i = Y(e_i)$.

Meta-Features. The first step to generate a meta-example from a problem is to extract its meta-features. In our work, a total number of $p = 10$ meta-features adopted in [14] was used to describe the datasets of regression problems:

1. X_1 - Log of the number of training examples;
2. X_2 - Log of the ratio between number of training examples and attributes;
3. X_3, X_4, X_5 and X_6 - Minimum, maximum, mean and standard deviation of the absolute values of correlation between predictor attributes and the target attribute;
4. X_7, X_8, X_9 and X_{10} - Minimum, maximum, mean and standard deviation of the absolute values of correlation between pairs of predictor attributes.

The meta-feature X_1 is an indicator of the amount of data available for training, and X_2 , in turn, indicates the dimensionality of the dataset. The meta-features X_3, X_4, X_5 and X_6 indicate the amount of relevant information available to predict the target attribute. The meta-features X_7, X_8, X_9 and X_{10} , in turn, indicate the amount of redundant information in the dataset.

Performance Information. The second step to generate a meta-example is to estimate the performance of the candidate algorithms on the problem being tackled. In our prototype, this step consists of evaluating the performance of BP and LM when used to train one-hidden layer MLPs¹. From this evaluation, we produce the performance information (Y) stored in the meta-examples which will correspond to the difference between the estimated performance of BP and LM. In order to measure the performance of each training algorithm in each problem, the following methodology of evaluation was applied.

The dataset was divided in the training, validation and test sets, in the proportion of 50%, 25% and 25%. As usual, the training set was used to adjust the MLP's weights, the validation set was used to estimate the MLP performance during training, and the test set was used to evaluate the performance of the trained MLP. The optimal number of hidden nodes was defined by testing the values 1, 2, 4, 8, 16 and 32. For each number of nodes, the MLP was trained 10 times with random initial weights. In the training process, we adopted benchmarking rules [16]: early stopping was used to avoid overfitting with the GL_5

¹ The BP and LM algorithms were implemented by using the NNET Matlab toolbox. Learning rates were defined by default.

stopping criterion and a maximum number of 1000 training epochs (see [16] for details of these rules). The optimal number of nodes was chosen as the value in which the MLP obtained the lowest average NMSE (Normalized Mean Squared Error) on the validation set over the 10 runs. The NMSE is defined as:

$$NMSE = \frac{\sum_{i=1}^{n_v} (t_i - o_i)^2}{\sum_{i=1}^{n_v} (t_i - \bar{t})^2} \quad (1)$$

In the equation, n_v is the number of examples in the validation set, t_i and o_i are respectively the true and the predicted value of the target attribute for example i , and \bar{t} is the average of the target attribute. The NMSE values have no scale and are comparable across different datasets, which is adequate to Meta-Learning [17]. Values of NMSE lower than 1 indicate that the MLP provided better predictions than the mean value at least.

The above methodology was applied to each learning algorithm (BP and LM). The performance information stored in the meta-example is the difference between the NMSE values obtained by the trained MLP (with optimal number of nodes) on the test subset, respectively using the BP and the LM algorithm.

3.2 Meta-Learner

The set of meta-examples is used to build a meta-regressor which will predict the performance measure Y for new problems. In our prototype, we deployed the Linear Regression (LR) model, which has some advantages to Meta-Learning. First, it produces in a short time a model which is easy to interpret [3]. Second, as it will be seen, an Active Learning method can be straightforwardly derived from the LR model using the confidence intervals estimated for its predictions.

The LR model is described in different textbooks (e.g., [18]). Here, we provide a short description of the LR model, which is adequate for understanding our work. Given an input problem e described by the vector $x = (x^1, \dots, x^p)$, the prediction $\hat{y} = \hat{Y}(e)$ provided by the meta-regressor can be defined as:

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j * x^j \quad (2)$$

The model may be more conveniently written in a matrix notation as:

$$\hat{y} = \mathbf{x}\hat{\beta} \quad (3)$$

where

$$\mathbf{x} = [1 \ x^1 \ x^2 \ \dots \ x^p]$$

and

$$\hat{\beta} = [\hat{\beta}_0 \ \hat{\beta}_1 \ \dots \ \hat{\beta}_p]^T.$$

The parameters represented in $\hat{\beta}$ are derived by the least-squares method which minimizes the sum of squared error for the training data. The least square estimator $\hat{\beta}$ is computed from the n training meta-examples in ME as:

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{y} \quad (4)$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^p \\ 1 & x_2^1 & x_2^2 & \dots & x_2^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n^1 & x_n^2 & \dots & x_n^p \end{bmatrix}$$

and

$$\mathbf{y} = [y_1 \ y_2 \ \dots \ y_n]^T.$$

3.3 Active Learning

As seen, the ML module generates a meta-regression model from a set of *labeled* meta-examples, associated to the set of problems E in which the performance information is known. The AL module receives a set of *unlabeled* problems \tilde{E} in which the algorithms were not yet evaluated. The AL module incrementally selects unlabeled problems in \tilde{E} to be used for generating new meta-examples.

In the prototype, the AL module deployed an uncertainty-based active learning method. In this approach, the Meta-Regressor initially generates its predictions for each unlabeled example $\tilde{e} \in \tilde{E}$ by using the estimated model (see section 3.2). A degree of prediction uncertainty $\mathcal{S}(\tilde{e}|E)$ is assigned to each unlabeled example. Finally, the unlabeled example with the highest prediction uncertainty is selected. The uncertainty-based approach can be directly applied to the LR model, since confidence intervals for its responses can be reliably estimated (see [18]). A $100(1 - \alpha)$ percent confidence interval of \hat{y} is defined as:

$$\hat{y} - t_{\alpha/2, n-p-1} \sqrt{V(\hat{y})} \leq y \leq \hat{y} + t_{\alpha/2, n-p-1} \sqrt{V(\hat{y})} \quad (5)$$

In the above equation, $V(\hat{y})$ is the variance of the prediction \hat{y} which is estimated as:

$$V(\hat{y}) = \hat{\sigma}^2 \mathbf{x}' (\mathbf{X}' \mathbf{X})^{-1} \mathbf{x} \quad (6)$$

where $\hat{\sigma}^2$ is the residual mean square of the LR model, defined as:

$$\hat{\sigma}^2 = \frac{\mathbf{y}^T \mathbf{y} - \hat{\beta}^T \mathbf{X}^T \mathbf{y}}{n - p - 1} \quad (7)$$

The size of the confidence interval defined in Eq. (5), which is $2 * t_{\alpha/2, n-p} \sqrt{V(\hat{y})}$, reflects the uncertainty of the prediction provided by the LR model. Since this value for different unlabeled examples solely depends on $V(\hat{y})$, we defined the degree of prediction uncertainty in the AL module as:

$$\mathcal{S}(\tilde{e}|E) = V(\hat{y}) \quad (8)$$

where \hat{y} is the prediction provided by the meta-regressor for the problem \tilde{e} . The AL module then selects, for generating a new meta-example, the problem $\tilde{e}^* \in \tilde{E}$ with highest uncertainty. A new meta-example is then generated from \tilde{e}^* as described in section 3.1.

4 Experiments and Results

In our work, we performed experiments with a set of meta-examples generated from 50 regression problems, available in the WEKA project². On average, the collected datasets presented 4,392 examples and 13.92 attributes. We observed in these datasets a large variability in both the number of examples and attributes, which is convenient to Meta-Learning studies.

A leave-one-out experiment was performed to evaluate the performance of the meta-regressor, also varying the number of meta-examples provided by the AL module. At each step of leave-one-out, one problem is left out for testing the ML module, and the remaining 49 problems are considered as candidates to generate meta-examples. The AL module progressively includes one meta-example in the training set of the ML module, up to the total number of 49 training meta-examples. At each included meta-example, the ML module is used to predict the algorithm performance on the test problem left out, and the obtained squared error is registered. Hence, a curve with 49 squared errors is produced for each test problem. Finally, the curve of error rates obtained by ML is summarized by computing the NMSE measure over the 50 steps of the leave-one-out experiment.

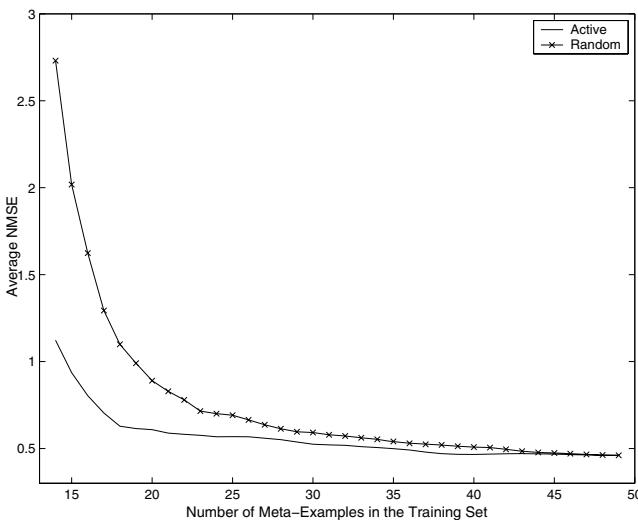


Fig. 2. Average curves of error rates for active and random methods. For simplicity, high rates ($NMSE > 3$) obtained in the early part of the curves were not displayed.

As a basis of comparison, the same above experiment was applied by using a random method for selecting unlabeled problems. Despite its simplicity, the random method has the advantage of performing a uniform exploration of the

² These regression datasets are specifically the sets provided in the files *numeric* and *regression* available in <http://www.cs.waikato.ac.nz/ml/weka/>

example space, and hence, it has been very competitive when compared to other active methods [19]. Finally, we highlight that the leave-one-out experiment was performed in 30 different runs for both the AL method using uncertainty sampling and the random procedure. The performance of the meta-regressor was evaluated by averaging the curves of error rates over the 30 runs.

Figure 2 presents the average curve of error rates for the active and the random methods. For both methods, the error rate obtained by the ML module decreased as the number of meta-examples in the training set increased. However, the error rates obtained by deploying the active method were, in general, lower than the error rates obtained by deploying the random method. From 17 meta-examples included in the training set, the NMSE values obtained by using the active method were lower than 1 (a threshold which indicates the viability of using the meta-regressor). By considering the random method, NMSE values lower than 1 were achieved only from 24 included meta-examples. In the experiments, we also compared the active method to the random method in statistical terms, by applying a t-test (95% of confidence) to the difference of error rates obtained by the meta-regressor in each point of the curves of error rates. In this evaluation, the active method obtained a statistical gain over the random method from 15 to 34 meta-examples included in the training set, which represent 40.82% of the 49 points considered in the curves.

5 Conclusion

In this paper, we presented the use of Active Learning to support the generation of training examples for Meta-Regression. In our proposal, Active Learning methods are used to reduce the cost of generating a good set of meta-examples while maintaining the performance of the meta-regressor. We highlight that our work brings contributions in both fields the Meta-Learning and the Active Learning.

In order to verify the viability of our proposal, we implemented a prototype to select meta-examples for a LR meta-regressor. The prototype was evaluated in a task of predicting the performance of MLPs, achieving satisfactory experimental results. Despite the advantages of LR as meta-learner, a large number of regression algorithms could be deployed. For instance, we intend to evaluate the use of Support Vector Machines (SVMs) as meta-regressors. Consequently, we will adapt active techniques previously proposed for standard SVMs (e.g., [20]). We also intend to propose active methods for other Meta-Learning approaches which include, for instance, ranking approaches (e.g., [8,9]).

Acknowledgments. The authors would like to thank CNPq (Brazilian Agency) for its support.

References

1. Giraud-Carrier, C., Vilalta, R., Brazdil, P.: Introduction to the special issue on meta-learning. *Machine Learning* 54(3), 187–193 (2004)
2. Vilalta, R., Giraud-Carrier, C., Brazdil, P., Soares, C.: Using meta-learning to support data-mining. *Intern. Journal of Computer Science Application* I(31), 31–45 (2004)

3. Gama, J., Brazdil, P.: Characterization of classification algorithms. In: Pinto-Ferreira, C., Mamede, N.J. (eds.) EPIA 1995. LNCS, vol. 990, pp. 189–200. Springer, Heidelberg (1995)
4. Bensusan, H., Alexandros, K.: Estimating the predictive accuracy of a classifier. In: Proc. of the 12th European Conference on Machine Learning, pp. 25–36 (2001)
5. Koepf, C.: Meta-regression: performance prediction. In: Meta-Learning: Strategies, Implementations, and Evaluations for Algorithm Selection, pp. 89–106 (2006)
6. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. *Machine Learning* 15, 201–221 (1994)
7. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* 323, 533–536 (1986)
8. Brazdil, P., Soares, C., da Costa, J.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning* 50(3), 251–277 (2003)
9. Kalousis, A., Gama, J., Hilario, M.: On data and algorithms - understanding inductive performance. *Machine Learning* 54(3), 275–312 (2004)
10. Aha, D.: Generalizing from case studies: A case study. In: Proceedings of the 9th International Workshop on Machine Learning, pp. 1–10. Morgan Kaufmann, San Francisco (1992)
11. Kalousis, A., Hilario, M.: Representational issues in meta-learning. In: Proceedings of the 20th International Conference on Machine Learning, pp. 313–320 (2003)
12. Prudêncio, R.B.C., Ludermir, T.B.: Meta-learning approaches to selecting time series models. *Neurocomputing* 61, 121–137 (2004)
13. Leite, R., Brazdil, P.: Predicting relative performance of classifiers from samples. In: Proceedings of the 22nd International Conference on Machine Learning (2005)
14. Prudêncio, R.B.C., Ludermir, T.B.: Selective generation of training examples in active meta-learning. *Intern. Journal of Hybrid Intelligent Systems* 5, 59–70 (2008)
15. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathematics* II(2), 164–168 (1944)
16. Prechelt, L.: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Universit   Karlsruhe, Germany (1994)
17. Soares, C., Brazdil, P., Kuba, P.: A meta-learning approach to select the kernel width in support vector regression. *Machine Learning* 54(3), 195–209 (2004)
18. Montgomery, D., Peck, E., Vining, G.: Introduction to Linear Regression Analysis, 4th edn. Wiley, Chichester (2006)
19. Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective sampling for nearest neighbor classifiers. *Machine Learning* 54, 125–152 (2004)
20. Schohn, G., Cohn, D.: Less is more: Active learning with support vector machines. In: Proc. of the 17th Intern. Conference on Machine Learning, pp. 839–846 (2000)

A Maximum-Likelihood Connectionist Model for Unsupervised Learning over Graphical Domains

Edmondo Trentin and Leonardo Rigutini

DII – Università di Siena, V. Roma, 56 Siena, Italy

Abstract. Supervised relational learning over labeled graphs, e.g. via recursive neural nets, received considerable attention from the connectionist community. Surprisingly, with the exception of recursive self organizing maps, unsupervised paradigms have been far less investigated. In particular, no algorithms for density estimation over graphs are found in the literature. This paper introduces first a formal notion of probability density function (pdf) over graphical spaces. It then proposes a maximum-likelihood pdf estimation technique, relying on the joint optimization of a recursive encoding network and a constrained radial basis functions-like net. Preliminary experiments on synthetically generated samples of labeled graphs are analyzed and tested statistically.

Keywords: Density estimation, unsupervised relational learning, recursive network.

1 Introduction

Two major instances of unsupervised learning have long been considered in statistical pattern recognition, namely the estimation of probability density functions (pdf), and clustering algorithms [2]. An approximative borderline between the two setups can be traced by saying that the former focuses on the probabilistic properties of the data sample, whilst the latter is rather topology-oriented, in the sense that it concentrates on certain topological properties (e.g., distance measures among patterns and/or centroids). Several unsupervised training algorithms for neural networks were also introduced (e.g., competitive neural nets and self-organizing maps). Most of these neural networks are rooted in the topological framework (i.e., clustering, topologically consistent mappings, etc.), although a few exceptions aimed at pdf estimation can be found [8].

It is rather surprising to realize that, despite the amount of work that has been accomplished in the community on relational and graphical learners in the last few decades, only limited attention has been paid to unsupervised relational learning (a remarkable exception, in the topological framework, is [5]), and to pdf estimation in the first place. This is even more surprising if we consider the fact that the original motivations for undertaking the study of unsupervised algorithms are often strong in the graphical domains (such as the World Wide Web, or as biological and chemical data which have a natural representation in

terms of variable-size labeled graphs), where amounts of unlabeled samples are available. These motivations include: (i) the need for a compact description of the overall distribution of a sample; (ii) the need for a measure of likelihood of a certain model given a graphical structure to be assigned to a certain group (“cluster”); (iii) the need for techniques that can deal with large amounts of unlabeled data in order to ease the design of semi-supervised classifiers or to facilitate the adaptation of previously trained machines to new data or new environmental conditions.

This paper is a first attempt to introduce a model for the estimation of pdfs over graphical domains. It exploits the encoding capabilities of recursive neural nets (RNN) [7], combined with a constrained radial basis function (RBF)-like network. A gradient ascent, maximum-likelihood (ML) training algorithm is proposed, which jointly optimizes the encoding network [7] and the RBF in order to obtain the pdf estimate from an unsupervised sample of graphs. Constraints are introduced in both neural nets such that the resulting estimate can be interpreted as a pdf (non-negative, unit integral over its definition domain), and such that the encoding of the graphs does not lead to singular solutions.

In order to introduce the model, it is necessary to give a formal definition of a pdf over graphs in the first place, via the notion of generalized random graph (an extension of traditional random graphs [3]). Let \mathcal{V} be a given discrete or continuous-valued set (*vertex universe*), and let Ω be any given sample space. We define a generalized random graph (GRG) over \mathcal{V} and Ω as a function $G : \Omega \rightarrow \{(V, E) | V \subseteq \mathcal{V}, E \subseteq V \times V\}$ (note that labels in the form of real-valued vectors associated with vertices and/or edges is easily encapsulated within the definition). Let then $\mathcal{G} = \{(V, E) | V \subseteq \mathcal{V}, E \subseteq V \times V\}$ be the space of GRG outcomes. We define a probability density function (*pdf*) for GRGs over \mathcal{V} as a function $p : \mathcal{G} \rightarrow \mathbb{R}$ such that (1) $p(g) \geq 0, \forall g \in \mathcal{G}$, and (2) $\int_{\mathcal{G}} p(g) dg = 1$. Note that the integral in (2) has a mathematical meaning since the (Lebesgue) measurability of the space of graphs defined over measurable domains (and with measurable labels) like countable sets or real vectors is shown in [4]. The extension of notions from traditional probability theory (conditional pdf, joint pdf, statistical independence) to GRGs is a straightforward exercise. Now, suppose that a sample $\mathcal{T} = \{g_1, \dots, g_n | g_i \in \mathcal{G}, i = 1, \dots, n\}$ of n graphs has been observed. The pdf estimation problem faced in this paper can be stated as follows: assuming that all the GRG outcomes in \mathcal{T} have been independently drawn from a certain pdf $p(g)$, how can the dataset be used in order to estimate a “reasonable” model of $p(g)$?

2 A Plausible Neural Answer to the Question

We assume that $p(g)$ is a function having fixed and known parametric form, being determined uniquely by the specific value of a set of *parameters* $\boldsymbol{\theta} = (\theta_1, \dots, \theta_k)$. To render this dependency on $\boldsymbol{\theta}$ in a more explicit manner, we will modify our notation slightly by writing $p(g)$ as $p(g|\boldsymbol{\theta})$. Given the assumption, the formulation of the question posed at the end of previous Section can be restated as: how can we use the sample \mathcal{T} in order to obtain estimates for $\boldsymbol{\theta}$ that

are meaningful according to a certain “optimality” criterion? A sound answer to the question may be found in the adoption of the ML criterion, along with a suitable method for maximizing the likelihood $p(\mathcal{T}|\boldsymbol{\theta})$ of the parameters given the sample. Since g_1, \dots, g_n are assumed to be i.i.d., the likelihood $p(\mathcal{T}|\boldsymbol{\theta})$ can be written as $p(\mathcal{T}|\boldsymbol{\theta}) = \prod_{i=1}^n p(g_i|\boldsymbol{\theta})$. Before attempting the maximization of the likelihood, it is necessary to specify a well-defined form for the pdf $p(g|\boldsymbol{\theta})$. Let us assume the existence of an integer d and of two functions, $\phi : \mathcal{G} \rightarrow \mathbb{R}^d$ and $\hat{p} : \mathbb{R}^d \rightarrow \mathbb{R}$, s.t. $p(g|\boldsymbol{\theta})$ can be decomposed as:

$$p(g|\boldsymbol{\theta}) = \hat{p}(\phi(g)). \quad (1)$$

It is seen that there exist (infinite) choices for $\phi(\cdot)$ and $\hat{p}(\cdot)$ that satisfy Eq. (1), the most trivial being $\phi(g) = p(g|\boldsymbol{\theta})$, $\hat{p}(x) = x$. We call $\phi(\cdot)$ the *encoding*, while $\hat{p}(\cdot)$ is simply referred to as the “likelihood”. Again, we assume parametric forms $\phi(g|\boldsymbol{\theta}_\phi)$ and $\hat{p}(\mathbf{x}|\boldsymbol{\theta}_{\hat{p}})$ for the encoding and for the likelihood, respectively, and we set $\boldsymbol{\theta} = (\boldsymbol{\theta}_\phi, \boldsymbol{\theta}_{\hat{p}})$. The ML estimation of $\boldsymbol{\theta}$ given \mathcal{T} requires now to find parameter vectors $\boldsymbol{\theta}_\phi$ and $\boldsymbol{\theta}_{\hat{p}}$ that maximize the quantity

$$p(\mathcal{T}|\boldsymbol{\theta}_\phi, \boldsymbol{\theta}_{\hat{p}}) = \prod_{i=1}^n \hat{p}(\phi(g_i|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}}). \quad (2)$$

We propose a two-block connectionist/statistical model for $p(g|\boldsymbol{\theta})$ as follows. The function $\phi(g|\boldsymbol{\theta}_\phi)$ is realized via an *encoding network*, suitable to map directed acyclic graphs (DAG) g into real vectors \mathbf{x} , as described in [7] for supervised training of recursive neural networks (RNN) over structured domains. The weights of the encoding network become the parameters $\boldsymbol{\theta}_\phi$. A radial basis functions (RBF)-like neural net is then used to model the likelihood function $\hat{p}(\mathbf{x}|\boldsymbol{\theta}_{\hat{p}})$, where $\boldsymbol{\theta}_{\hat{p}}$ are the parameters of the RBF. In order to ensure that a pdf is obtained, specific constraints have to be placed on the nature of the RBF kernels, as well as on the hidden-to-output connection weights. It is crucial to underline that we are not going to find out a rough encoding of graphs via standard RNN followed by a separate, standard ML estimation of a mixture of Normal densities defined over the encoded space. On the contrary, we propose a joint optimization of all model parameters, $\boldsymbol{\theta}_\phi$ and $\boldsymbol{\theta}_{\hat{p}}$, to increase the overall likelihood. In other words, the encoding and the likelihood are jointly optimized to maximize $p(\mathcal{T}|\boldsymbol{\theta}_\phi, \boldsymbol{\theta}_{\hat{p}})$. Occasionally, the ML principle for this general class of mixtures may lead to singular solutions. This fact is well-known from classical statistical theory; but, as pointed out in [2] (sec. 6.4.3., page 199), “it is an empirical fact that meaningful solutions can still be obtained”.

A hill-climbing algorithm to carry out ML estimation of the parameters $\boldsymbol{\theta}$ can be obtained as an instance of the gradient-ascent method over $p(\mathcal{T}|\boldsymbol{\theta}_\phi, \boldsymbol{\theta}_{\hat{p}})$ in two steps: (i) *initialization*, i.e., start with some initial, e.g. “random”, assignment of values to the model parameters $\boldsymbol{\theta}$; (ii) *gradient-ascent*, i.e., repeatedly apply a learning rule in the form $\Delta\boldsymbol{\theta} = \eta \nabla_{\boldsymbol{\theta}} \{ \prod_{i=1}^n \hat{p}(\phi(g_i|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}}) \}$ with $\eta \in \mathbb{R}^+$. This is a batch learning setup. In practice, neural network learning may be

simplified, yet even improved, with the adoption of an on-line training scheme that prescribes $\Delta\theta = \eta \nabla_{\theta} \{\hat{p}(\phi(g|\theta_{\phi})|\theta_{\hat{p}})\}$ upon presentation of each individual training example g . Three distinct families of adaptive parameters θ have to be considered:

(1) Mixing parameters c_1, \dots, c_n , i.e. the hidden-to-output weights of the RBF network. Constraints have to be placed on these parameters during the ML estimation process, in order to ensure that they are in $[0, 1]$ and that they sum to one. A simple way to satisfy the requirements is to introduce n hidden parameters $\gamma_1, \dots, \gamma_n$, which are unconstrained, and to set

$$c_i = \frac{\varsigma(\gamma_i)}{\sum_{j=1}^n \varsigma(\gamma_j)}, i = 1, \dots, n \quad (3)$$

where $\varsigma(x) = 1/(1 + e^{-x})$. Each γ_i is then treated as an unknown parameter θ to be estimated via ML.

(2) d -dimensional mean vector μ_i and $d \times d$ covariance matrix Σ_i for each of the Gaussian kernels $K_i(\mathbf{x}) = N(\mathbf{x}; \mu_i, \Sigma_i)$, $i = 1, \dots, n$ of the RBF, where $N(\mathbf{x}; \mu_i, \Sigma_i)$ denotes a multivariate Normal pdf having mean vector μ_i , covariance matrix Σ_i , and evaluated over the random vector \mathbf{x} . A common (yet effective) simplification is to consider diagonal covariance matrices, i.e. independence among the components of the input vector \mathbf{x} . This assumption leads to the following three major consequences: (i) modeling properties are not affected significantly, according to [6]; (ii) generalization capabilities of the overall model may turn out to be improved, since the number of free parameters is reduced; (iii) i -th multivariate kernel K_i may be expressed in the form of a product of d univariate Normal densities as:

$$K_i(\mathbf{x}) = \prod_{j=1}^d \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp \left\{ -\frac{1}{2} \left(\frac{x_j - \mu_{ij}}{\sigma_{ij}} \right)^2 \right\} \quad (4)$$

i.e., the free parameters to be estimated are the means μ_{ij} and the standard deviations σ_{ij} , for each kernel $i = 1, \dots, n$ and for each component $j = 1, \dots, d$ of the input space.

(3) The weights \mathcal{U} of the encoding network. The learning rule has to rely on partial derivatives of the likelihood which are backpropagated down to the RBF inputs and, in turn, through the encoding net. In order to discourage singular solutions, e.g. the tendency to map all the input graphs onto a single point in the encoded space by developing close-to-zero weights, the learning rule for \mathcal{U} shall include an additional regularization term which treats the network weights as random variables distributed according to a pdf whose modes are far from zero. The likelihood of the network weights is then taken into account in the optimization procedure.

In the following, we will derive explicit formulations for $\frac{\partial \hat{p}(\phi(g|\theta_{\phi})|\theta_{\hat{p}})}{\partial \theta}$ for each of the three families of free parameters θ within the proposed model. As regards a generic mixing parameter $c_i, i = 1, \dots, n$, from Eq. (3), and since $p(g) = \sum_{k=1}^n c_k K_k(\mathbf{x})$, we have

$$\begin{aligned}
\frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial \gamma_i} &= \sum_{j=1}^n \frac{\partial p(g)}{\partial c_j} \frac{\partial c_j}{\partial \gamma_i} \\
&= \sum_{j=1}^n K_j(\mathbf{x}) \frac{\partial}{\partial \gamma_i} \left(\frac{\varsigma(\gamma_j)}{\sum_{k=1}^n \varsigma(\gamma_k)} \right) \\
&= K_i(\mathbf{x}) \left\{ \frac{\varsigma'(\gamma_i) \sum_k \varsigma(\gamma_k) - \varsigma(\gamma_i) \varsigma'(\gamma_i)}{[\sum_k \varsigma(\gamma_k)]^2} \right\} + \sum_{j \neq i} K_j(\mathbf{x}) \left\{ \frac{-\varsigma(\gamma_j) \varsigma'(\gamma_i)}{[\sum_k \varsigma(\gamma_k)]^2} \right\} \\
&= K_i(\mathbf{x}) \frac{\varsigma'(\gamma_i)}{\sum_k \varsigma(\gamma_k)} - \sum_j K_j(\mathbf{x}) \frac{\varsigma(\gamma_j) \varsigma'(\gamma_i)}{[\sum_k \varsigma(\gamma_k)]^2} \\
&= K_i(\mathbf{x}) \frac{\varsigma'(\gamma_i)}{\sum_k \varsigma(\gamma_k)} - \left\{ \sum_j c_j K_j(\mathbf{x}) \right\} \frac{\varsigma'(\gamma_i)}{\sum_k \varsigma(\gamma_k)} \\
&= \frac{\varsigma'(\gamma_i)}{\sum_k \varsigma(\gamma_k)} \{K_i(\mathbf{x}) - p(g)\}.
\end{aligned} \tag{5}$$

For the means μ_{ij} and the standard deviations σ_{ij} we proceed as follows. Let θ_{ij} denote the free parameter, i.e. μ_{ij} or σ_{ij} , to be estimated. It is seen that:

$$\frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial \theta_{ij}} = c_i \frac{\partial K_i(\mathbf{x})}{\partial \theta_{ij}} \tag{6}$$

where the calculation of $\frac{\partial K_i(\mathbf{x})}{\partial \theta_{ij}}$ can be accomplished as follows. First of all, let us observe that for any real-valued, differentiable function $f(\cdot)$ this property holds true: $\frac{\partial f(\cdot)}{\partial x} = f(\cdot) \frac{\partial \log[f(\cdot)]}{\partial x}$. As a consequence, from Eq. (4) we can write

$$\begin{aligned}
\frac{\partial K_i(\mathbf{x})}{\partial \theta_{ij}} &= K_i(\mathbf{x}) \frac{\partial \log K_i(\mathbf{x})}{\partial \theta_{ij}} \\
&= K_i(\mathbf{x}) \frac{\partial}{\partial \theta_{ij}} \sum_{k=1}^d \left\{ -\frac{1}{2} \left[\log(2\pi\sigma_{ik}^2) + \left(\frac{x_k - \mu_{ik}}{\sigma_{ik}} \right)^2 \right] \right\}.
\end{aligned} \tag{7}$$

For the means, i.e. $\theta_{ij} = \mu_{ij}$, Eq. (7) yields

$$\frac{\partial K_i(\mathbf{x})}{\partial \mu_{ij}} = K_i(\mathbf{x}) \frac{x_j - \mu_{ij}}{\sigma_{ij}^2}. \tag{8}$$

For the covariances, i.e. $\theta_{ij} = \sigma_{ij}$, Eq. (7) takes the form:

$$\begin{aligned}
\frac{\partial K_i(\mathbf{x})}{\partial \sigma_{ij}} &= K_i(\mathbf{x}) \frac{\partial}{\partial \sigma_{ij}} \left\{ -\frac{1}{2} \log(2\pi\sigma_{ij}^2) - \frac{1}{2} \left(\frac{x_j - \mu_{ij}}{\sigma_{ij}} \right)^2 \right\} \\
&= \frac{K_i(\mathbf{x})}{\sigma_{ij}} \left\{ \left(\frac{x_j - \mu_{ij}}{\sigma_{ij}} \right)^2 - 1 \right\}.
\end{aligned} \tag{9}$$

Finally, let us consider the connection weights $\mathcal{U} = \{v_1, \dots, v_s\}$ within the encoding network. For a generic $v \in \mathcal{U}$, application of the chain rule yields:

$$\frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial v} = \frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial y} \frac{\partial y}{\partial v} \tag{10}$$

where y is the output from the unit (in the encoding net) which is fed from connection v . The quantity $\frac{\partial y}{\partial v}$ can be easily computed by taking the partial derivative of the activation function associated with the unit itself, as usual. As regards the quantity $\frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial y}$, we proceed as follows. First of all, let us assume that v feeds the output layer, i.e. it connects a certain hidden unit with j -th output unit of the encoding net. In this case, we have $y = x_j$. It is easy to see that:

$$\begin{aligned}\frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial x_j} &= \frac{\partial \sum_{i=1}^n c_i K_i(\mathbf{x})}{\partial x_j} \\ &= \sum_{i=1}^n c_i K_i(\mathbf{x}) \frac{\partial \log K_i(\mathbf{x})}{\partial x_j} \\ &= \sum_{i=1}^n c_i K_i(\mathbf{x}) \frac{\partial}{\partial x_j} \sum_{k=1}^d \left\{ -\frac{1}{2} \left[\log(2\pi\sigma_{ik}^2) + \left(\frac{x_k w_{ik} - \mu_{ik}}{\sigma_{ik}} \right)^2 \right] \right\} \\ &= \sum_{i=1}^n c_i K_i(\mathbf{x}) \left\{ -\frac{1}{2} \frac{\partial}{\partial x_j} \left(\frac{x_j w_{ij} - \mu_{ij}}{\sigma_{ij}} \right)^2 \right\} \\ &= -\sum_{i=1}^n c_i \frac{K_i(\mathbf{x})}{\sigma_{ij}^2} (x_j w_{ij} - \mu_{ij}) w_{ij}.\end{aligned}\tag{11}$$

On the contrary, whenever v is a hidden weight the quantity $\frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial v}$ can be obtained applying the usual *backpropagation through structures* (BPTS) algorithm [7], once the *deltas* to be backpropagated have been initialized at the output layer via Eq. (11). Unconstrained ML training of the weights of the encoding net may lead to singular solutions. To tackle the problem, we assume that the weights are random variables, independently drawn from a certain probability distribution $p(\mathcal{U}) = \prod_{i=1}^s p(v_i)$. The pdf $p(v)$ is defined in a way to encourage non-degenerate solutions, and the new criterion function C to be maximized during gradient-ascent training is in the form of a joint pdf, namely:

$$C = \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})p(\mathcal{U}).\tag{12}$$

Extremization of such a criterion results in weight values that yield high likelihood of the sample, and that are highly likely themselves. If the weights \mathcal{U} are randomly initialized in a uniform manner over the interval $(-\rho, \rho)$, an effective choice for $p(v)$ is a mixture of two Gaussian components in the form

$$p(v) = \frac{1}{2}N(v; -\frac{\rho}{2}, \sigma^2) + \frac{1}{2}N(v; \frac{\rho}{2}, \sigma^2).\tag{13}$$

Whenever σ^2 is chosen to be sufficiently small, two benefits are expected as training proceeds: (1) weights are encouraged to move toward non-degenerative solutions in the weight space; (2) a form of regularization of the learning process emerges, since complex solutions (i.e., weights too large in size) are discouraged.

Given a generic weight $v \in \mathcal{U}$, gradient ascent requires to compute partial derivatives of the proposed criterion C w.r.t. v , i.e.,

$$\frac{\partial C}{\partial v} = p(\mathcal{U}) \frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial v} + \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}}) \frac{\partial p(\mathcal{U})}{\partial v}. \quad (14)$$

The quantity $\frac{\partial \hat{p}(\phi(g|\boldsymbol{\theta}_\phi)|\boldsymbol{\theta}_{\hat{p}})}{\partial v}$ is computed as above, while the term $\frac{\partial p(\mathcal{U})}{\partial v}$ in Eq. (14) can be rewritten as follows:

$$\begin{aligned} \frac{\partial p(\mathcal{U})}{\partial v} &= p(\mathcal{U}) \frac{\partial \log p(\mathcal{U})}{\partial v} \\ &= p(\mathcal{U}) \frac{\partial}{\partial v} \sum_{i=1}^s \log p(v_i) \\ &= \frac{p(\mathcal{U})}{p(v)} \frac{\partial p(v)}{\partial v} \end{aligned} \quad (15)$$

which is computed in a straightforward manner by taking the derivatives of Eq. (13) w.r.t. v .

3 Demonstration

Since there are no other approaches to pdf estimation over graphical domains, comparisons are impracticable. Consequently, we analyze the behavior of the model and we evaluate it via statistical tests on a synthetic task. Two different samples of GRGs were synthesized under controlled probabilistic conditions. A first sample of 300 independent DAGs was randomly generated. These DAGs had a random number of vertices (between 5 and 20), a uniform distribution of edges connectivity (as in the classic Erdős and Rényi model [3]), and a real-valued label for vertices drawn from the Laplacian pdf $\frac{1}{2\theta} \exp\left(-\frac{|x-\mu|}{\theta}\right)$ with location $\mu = 5.0$ and smoothness $\theta = \frac{1}{2}$. Let us call \mathcal{Q} this collection of GRGs. \mathcal{Q} was partitioned into three equally-sized subsamples, \mathcal{Q}_0 (training set), \mathcal{Q}_1 (validation set) and \mathcal{Q}_2 (test set).

Another sample \mathcal{P} of 200 independent DAGs was likewise obtained, each DAG having: a random number of vertices (between 5 and 20), a Power-law “preferential attachment” random connectivity (as in Barabási and Albert model [1]) according to the value of the node labels (the relevance, or “authority”), which were independently drawn from an exponential distribution $\lambda e^{-\lambda x}$, with inverse scale $\lambda = \frac{1}{3}$. The collection \mathcal{P} was split into equally-sized subsamples, \mathcal{P}_1 (reference set) and \mathcal{P}_2 (reference test set), as well.

The pdf underlying the distributions of GRGs in \mathcal{Q} was then estimated, relying on the training subsample \mathcal{Q}_0 . An encoding net having 10 sigmoid hidden units and 2 linear encoding neurons was used, while the RBF used 4 Gaussian kernels. All the parameters were initialized at random. Different learning rates (chosen, along with the neural architectures, by evaluating the variations of the likelihood on training and validation sets) were applied for the different families of parameters, namely $\eta_\gamma = 1.0 \cdot 10^{-4}$, $\eta_\mu = 5.0 \cdot 10^{-5}$, $\eta_\sigma = 5.0 \cdot 10^{-6}$,

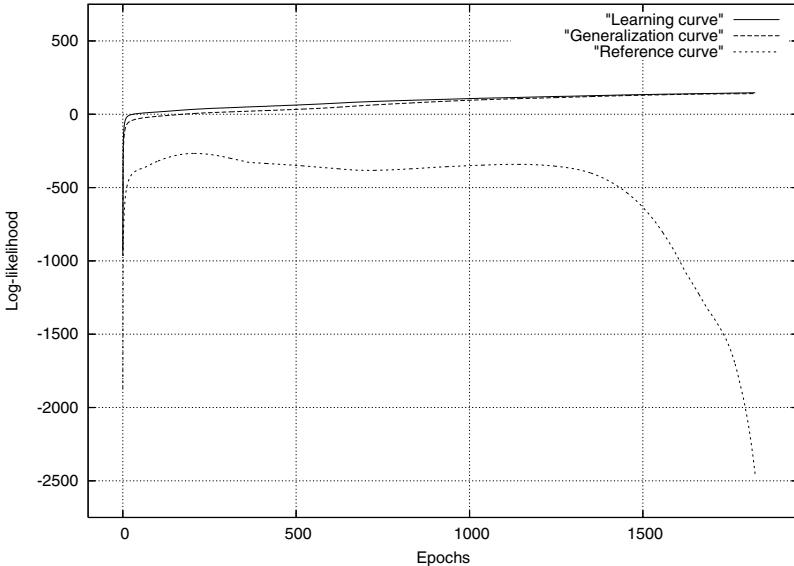


Fig. 1. Learning, generalization and reference curves

$\eta_v = 1.0 \cdot 10^{-7}$ (the notation implicitly refers to the symbols used in Section 2). Fig. 1 shows the learning curve (log-likelihood on \mathcal{Q}_0), generalization curve (log-likelihood on \mathcal{Q}_1), and reference curve (log-likelihood on \mathcal{P}_1). It is seen that the criterion function is increased during training, as expected. All three curves exhibit a steep growth during the early training, due to the fact that the RBF kernels quickly move from their random initial position toward the region in \mathbb{R}^2 where all the graphs are initially randomly encoded. Learning and generalization curves continue to grow smoothly, getting closer to each other. This fact, magnified in Fig. 2 (left), indicates that the estimated pdf model explains (i.e., has high likelihood) equally well independent samples of GRGs drawn from the same distribution. On the contrary, the reference curve is constantly and significantly lower than the others, and starts dropping early (i.e., the model does not cover samples drawn from a different pdf). This is due to the constrained training of the RBF (which is forced to have a unit integral over its definition domain, i.e. it peaks around the GRGs in \mathcal{Q} at the expense of those in \mathcal{P}), and to the regularized training of the encoding network (whose weights are discouraged to move toward solutions that could map all the GRGs onto a compact cluster, regardless of their original distribution). Early stopping of training was accomplished once the generalization curve began decreasing (after 1824 iterations), whilst the learning curve continued to grow (overfitting the training data). This is seen also in Fig. 2 (right), which plots the difference between the two curves, which lowers down to a minimum at epoch 1824 before inverting its trend. The final average log-likelihood over \mathcal{Q}_0 , \mathcal{Q}_2 and \mathcal{P}_2 , respectively, is shown in Fig. 3 as a function of the number of GRGs in the training set (results are averaged w.r.t.

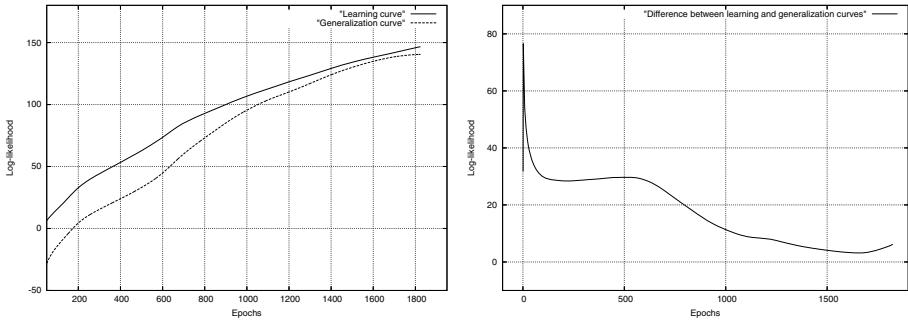


Fig. 2. Learning and generalization curves magnified (left). Difference between learning and generalization curves (right).

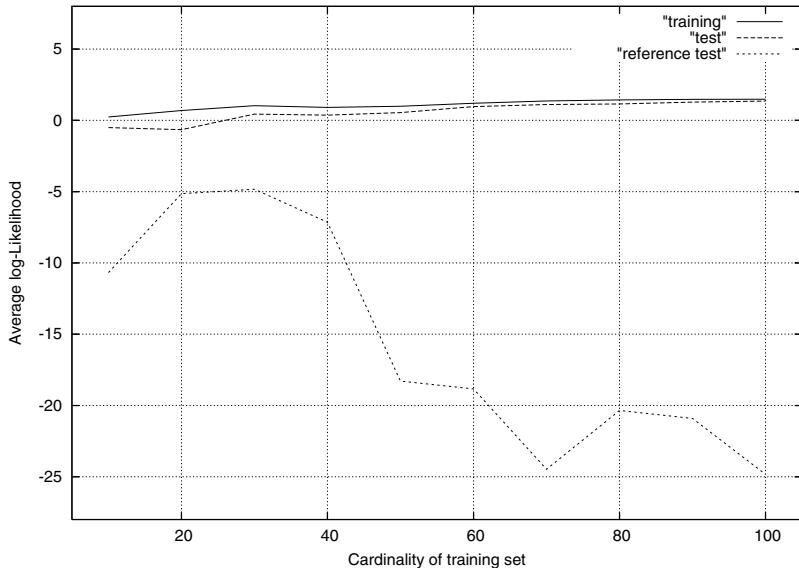


Fig. 3. Final average log-Likelihoods as a function of the training set size

the cardinality of the training set; the experiment was repeated, accordingly, for the different cardinalities).

Let us now call $\tilde{p}(\cdot)$ the pdf model estimated from \mathcal{Q}_0 (i.e., using 100 GRGs). Its capability to describe the statistical properties of the corresponding distribution \mathcal{Q} (but not those of the other, \mathcal{P}) may be quantified by evaluating how likely it explains the test samples \mathcal{Q}_2 and \mathcal{P}_2 , according to some statistical criteria. First of all, in the spirit of the likelihood-ratio test, the overall log-likelihoods $\mathcal{L}(\mathcal{Q}_2)$ and $\mathcal{L}(\mathcal{P}_2)$ of the model given \mathcal{Q}_2 and \mathcal{P}_2 , respectively, were computed. Let us define $\ell(g) = \log \tilde{p}(g)$ for any given graph g , and let $\Lambda = \log(\mathcal{L}(\mathcal{Q}_2)/\mathcal{L}(\mathcal{P}_2))$ be the (log)likelihood-ratio. Table II reports the statistics.

Table 1. Log-Likelihoods and Likelihood-ratio (Λ) tests of the estimated pdf model

$\mathcal{L}(\mathcal{Q}_2) = \sum_{g \in \mathcal{Q}_2} \ell(g)$	$\mathcal{L}(\mathcal{P}_2) = \sum_{g \in \mathcal{P}_2} \ell(g)$	Λ
134.21	-1897.21	2031.41

Roughly speaking, the model is highly likely to express the probabilistic law underlying \mathcal{Q} (but not \mathcal{P}), as sought. The value of Λ (i.e., the likelihood-ratio is $>> 1$) confirms the high statistical significance of the test. These values express global statistics. Let us now evaluate the distribution of individual log-likelihoods yielded by $\tilde{p}(.)$ over each graph in the test samples \mathcal{Q}_2 and \mathcal{P}_2 (100 values for each subsample) from an analytical point of view. To this end, the Kolmogorov-Smirnov (KS) test is a popular choice for the evaluation of pdf models. Two (independent) null-hypotheses were formed, namely: (1) “the distribution of individual log-likelihoods yielded by $\tilde{p}(.)$ when applied to \mathcal{Q}_2 coincides with the analogous distribution yielded by the same model on \mathcal{P}_2 ”. (2) “the distributions of individual log-likelihoods evaluated via $\tilde{p}(.)$ on the samples \mathcal{Q}_0 and \mathcal{Q}_2 do not coincide”. The KS test pointed out that both null-hypotheses are rejected at a level α of at least 0.001 (confidence $\geq 99.9\%$). That is, the model explains well the distribution of independent samples drawn from \mathcal{Q} , but is highly unlikely to explain GRGs having a different underlying distribution.

4 Conclusion and On-Going Work

This paper was a first attempt to introduce pdf estimation over graphical domains. It gave a formal notion of GRG, and proposed a combined connectionist model with joint, gradient-ascent constrained optimization of the parameters over the ML criterion. The model was evaluated in terms of statistical tests (likelihood-ratio, KS) on synthetic distributions of GRGs. On-going work focuses on applications to real-world tasks (e.g., classification and clustering of relational data).

References

1. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. *Science* 286, 509–512 (1999)
2. Duda, R., Hart, P.: *Pattern Classification and Scene Analysis*. Wiley, N.Y. (1973)
3. Erdős, P., Rényi, A.: On random graphs. *Publ. Math. Debrecen* 6, 290–297 (1959)
4. Hammer, B., Micheli, A., Sperduti, A.: Universal approximation capability of cascade correlation for structures. *Neural Computation* 17(5), 1109–1159 (2005)
5. Hammer, B., Micheli, A., Sperduti, A., Strickert, M.: Recursive self-organizing network models. *Neural Networks* 17(8-9), 1061–1085 (2004)
6. McLachlan, G.J., Basford, K.E. (eds.): *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York (1988)
7. Sperduti, A., Starita, A.: Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks* 8(3), 714–735 (1997)
8. Trentin, E., Gori, M.: Robust combination of neural networks and hidden Markov models for speech recognition. *IEEE Trans. on Neural Networks* 14(6) (2003)

Local Feature Selection for the Relevance Vector Machine Using Adaptive Kernel Learning

Dimitris Tzikas¹, Aristidis Likas¹, and Nikolaos Galatsanos²

¹ Department of Computer Science, University of Ioannina, Ioannina, Greece, 45110
{tzikas,arly}@cs.uoi.gr

² Department of Electrical Engineering, University of Patras, Rio, Greece, 26500
ngalatsanos@upatras.gr

Abstract. A Bayesian learning algorithm is presented that is based on a sparse Bayesian linear model (the Relevance Vector Machine (RVM)) and learns the parameters of the kernels during model training. The novel characteristic of the method is that it enables the introduction of parameters called ‘scaling factors’ that measure the significance of each feature. Using the Bayesian framework, a sparsity promoting prior is then imposed on the scaling factors in order to eliminate irrelevant features. Feature selection is local, because different values are estimated for the scaling factors of each kernel, therefore different features are considered significant at different regions of the input space. We present experimental results on artificial data to demonstrate the advantages of the proposed model and then we evaluate our method on several commonly used regression and classification datasets.

1 Introduction

In supervised learning we are given a training set $\{\mathbf{x}_n, t_n\}_{n=1}^N$, so that t_n is a noisy measurement of the output of a function y when its input is \mathbf{x}_n . Then, we wish to predict the output $y(\mathbf{x})$ of the function at any arbitrary test point \mathbf{x} . In regression the outputs t_n are continuous and they usually contain additive noise ϵ_n :

$$t_n = y(\mathbf{x}_n) + \epsilon_n, \quad (1)$$

while in classification the outputs t_n are discrete and assuming K classes they can be coded so that $t_{nk} = 1$ if x_n belongs to class k , otherwise $t_{nk} = 0$.

In order to make predictions a specific parametric form may be assumed for the unknown function y , such as a linear model:

$$y(\mathbf{x}|\mathbf{w}) = \sum_{i=1}^M w_i \phi_i(\mathbf{x}), \quad (2)$$

where $\mathbf{w} = (w_1, \dots, w_M)^T$ are the weights of the linear model and $\{\phi_i(\mathbf{x})\}_{i=1}^M$ is the set of basis functions, which are assumed fixed and must be selected a priori. The sparse Bayesian linear model [2] is a Bayesian treatment of the linear

model, which assumes a Gaussian prior with separate precision variable α_i for each weight w_i

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^M N(w_i|0, \alpha_i^{-1}), \quad (3)$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)^T$. Using this prior, many of the available basis functions are pruned during learning. Because of this, we can design linear models that initially assume a large number of basis functions and the final estimation will use only a small number of the available basis functions. For example, the relevance vector machine (RVM) is a specific instance of the sparse Bayesian linear model, which assumes that the basis functions are kernels and one kernel is placed at each training example.

An adaptive kernel learning methodology [1] has been recently proposed that automatically learns parameters of the basis functions of a sparse linear model. More specifically, it assumes that *each basis function has different parameters*, and in principle it can even have different parametric form, therefore it is very flexible. In order to avoid overfitting a sparsity enforcing prior has been used for the weight precisions $\boldsymbol{\alpha}$ that directly *controls the number of effective parameters* of the model [3]. Learning is then achieved using an algorithm that is similar to the incremental RVM algorithm [4]. It starts with an empty model and at each iteration it adds to the model an appropriate basis function, in order to maximize the marginal likelihood of the model. In the typical incremental RVM [4] selecting a basis function is achieved using discrete optimization over the location of the basis functions; all candidate basis functions are tested for addition to the model. In contrast, the adaptive kernel learning methodology [1] uses *continuous optimization* with respect to the parameters (such as location and scale) of the basis functions. This methodology has been applied to learn the center (mean) and width (variance) parameters of Gaussian kernel basis functions.

In supervised learning problems, feature selection is typically performed as a preprocessing step, which is performed before building a classification or regression model. The general idea is to eliminate irrelevant features in the training set, in order to improve the generalization performance of the model and simultaneously reduce the computational cost of its training. However, it is possible to design supervised learning models that incorporate feature selection mechanisms, in order to perform feature selection simultaneously with estimation of model parameters. These models need to consider all the available features for training and, for this reason, they have relatively high computational cost. However, they can achieve better performance in feature selection, because they can exploit information that the trained model provides. For example, in [5] the JCFO classification method is suggested that jointly selects relevant features and estimates parameters of the classifier. The classifier that they use is based on a linear model and feature selection is achieved by estimating parameters of the kernel function. More specifically, a *scaling factor* h_i is estimated for each feature x_i , which measures the significance of that feature. For example, in the case of *anisotropic* Gaussian kernels, the scaling factors correspond to the

inverse variance of each feature. Then a Laplacian sparsity prior is enforced on the scaling factors $\mathbf{h} = (h_1, \dots, h_d)^T$ in order to eliminate irrelevant features.

In this work we propose a Bayesian method to incorporate a feature selection mechanism in the adaptive kernel learning approach for the RVM (called aRVM) proposed in [1]. This method is similar in spirit to JCFO in that they both estimate parameters of kernels that are called scaling factors in order to measure the significance of each feature. However, the proposed approach (that is based on aRVM) learns *separate scaling factors for each kernel*, therefore feature selection is *local*, since it is performed for each kernel separately. This might be useful for example when different features are significant for discriminating examples of each class, as demonstrated in the example of Fig. [1].

The rest of this paper is organized as follows. In Section 2 we present an overview of sparse Bayesian linear models for regression and classification problems. In Section 3 we propose a method to incorporate local feature selection to the sparse Bayesian linear model, by adapting the kernel learning algorithm of [1]. In Section 4 we first present an artificial example to demonstrate the advantages of the proposed method and then we evaluate its performance on regression and classification datasets.

2 Sparse Bayesian Linear Models

2.1 Sparse Bayesian Regression

Since RVM is a Bayesian linear model, in this section we will review sparse Bayesian learning of linear models given by [2] [3]. We assume Gaussian distributed noise with separate precision β_n for each data point $p(\epsilon_n|\beta_n) = N(\epsilon_n|0, \beta_n^{-1})$, and a Gaussian prior with separate variance for the weights given by [3].

Defining the fixed ‘design’ matrix $\Phi = (\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_M))^T$, with $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_M(\mathbf{x}))^T$, the likelihood can be written as:

$$p(\mathbf{t}|\mathbf{w}, \boldsymbol{\beta}) = N(\mathbf{t}|\Phi\mathbf{w}, \mathbf{B}^{-1}), \quad (4)$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_N)^T$ and $\mathbf{B} = \text{diag}(\boldsymbol{\beta})$.

The posterior distribution of the weights can be computed using Bayes’s law:

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\mathbf{t}|\mathbf{w}, \boldsymbol{\beta})p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})}, \quad (5)$$

where $p(\mathbf{w}|\boldsymbol{\alpha})$ is given by [3]. It can be shown that the weight posterior distribution is given by [2]:

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = N(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad (6)$$

where

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\Phi^T\mathbf{B}\mathbf{t}, \quad (7)$$

$$\boldsymbol{\Sigma} = (\Phi^T\mathbf{B}\Phi + \mathbf{A})^{-1}, \quad (8)$$

and $\mathbf{A} = \text{diag}(\boldsymbol{\alpha})$.

In order to control the amount of sparsity, we use a prior on $\boldsymbol{\alpha}$ that directly penalizes models with large number of effective parameters [3]. The output of the model at the training points $\mathbf{y} = (y(\mathbf{x}_1), \dots, y(\mathbf{x}_N))^T$ can be evaluated as $\mathbf{y} = \mathbf{S}\mathbf{t}$, where $\mathbf{S} = \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{B}$ is the so called smoothing matrix. The ‘degrees of freedom’ of \mathbf{S} , given by $DF = \text{trace}(\mathbf{S})$, measure the effective number of parameters of the model. This motivates the following sparsity prior [3]:

$$p(\boldsymbol{\alpha}|\boldsymbol{\beta}) \propto \exp(-c \text{trace}(\mathbf{S})), \quad (9)$$

where the parameter c provides a mechanism to control the amount of desired sparsity. When using specific values of the sparsity parameter c , some known model selection criteria are obtained [6]:

$$c = \begin{cases} 0 & \text{None (typical RVM),} \\ 1 & \text{AIC (Akaike information criterion),} \\ \log(N)/2 & \text{BIC (Bayesian information criterion),} \\ \log(N) & \text{RIC (Risk inflation criterion).} \end{cases} \quad (10)$$

Update formulas for the weight precisions $\boldsymbol{\alpha}$ can be obtained by maximizing the posterior distribution $p(\boldsymbol{\alpha}|\mathbf{t}, \boldsymbol{\beta}) \propto p(\mathbf{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})p(\boldsymbol{\alpha})$ [1]:

$$\alpha_i = \frac{\gamma_i}{\mu_i^2 - 2c\gamma_i\Sigma_{ii}}, \quad (11)$$

where $\gamma_i = 1 - \alpha_i\Sigma_{ii}$.

Furthermore, since the noise is assumed i.i.d., then $\mathbf{B} = \beta\mathbf{I}$ with $\beta = \beta_1 = \dots = \beta_N$ and we can also update the noise precision β . Assuming an uninformative prior $p(\beta)$ we only need to maximize the marginal likelihood $p(\mathbf{t}|\boldsymbol{\alpha}, \beta)$. Setting its derivative to zero, gives the equation

$$\frac{1}{2} \left[\frac{N}{\beta} - \|\mathbf{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 - \text{trace}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) \right] - \beta c \text{trace}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) = 0, \quad (12)$$

which can be easily solved numerically, in order to update the noise precision β .

2.2 Sparse Bayesian Classification

For simplicity we only consider binary classification and assume that the outputs are coded so that $t_n \in \{0, 1\}$ ¹. Then, the likelihood is given by:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}, \quad (13)$$

where $y_n = \sigma(y(\mathbf{x}_n|\mathbf{w}))$. Using the Laplacian approximation, the classification problem can be mapped to a regression problem [2] with heteroscedastic noise $p(\epsilon_n) = N(\epsilon_n|0, \beta_n)$. The noise precision is given by:

$$\beta_n = y_n(1 - y_n), \quad (14)$$

¹ Multiclass problems can be solved using the one-vs-all approach, which builds only two-class models.

and the regression targets $\hat{\mathbf{t}} = (\hat{t}_1, \dots, \hat{t}_N)^T$ are:

$$\hat{\mathbf{t}} = \Phi \mathbf{w} + \mathbf{B}^{-1}(\mathbf{t} - \mathbf{y}), \quad (15)$$

where $\mathbf{y} = (y_1, \dots, y_N)^T$ and $\mathbf{B} = \text{diag}(\beta_1, \dots, \beta_N)$.

3 A Bayesian Model for Local Feature Selection

Consider now the more general form of the sparse Bayesian linear model proposed in [1]:

$$y(\mathbf{x}) = \sum_{n=1}^M w_n \phi(\mathbf{x}; \boldsymbol{\theta}_n), \quad (16)$$

which assumes that all the basis functions have the same parametric form $\phi(\mathbf{x}; \boldsymbol{\theta}_n)$, but assumes separate parameter vector $\boldsymbol{\theta}_n$ for each basis function. This model is called adaptive RVM (aRVM). For example, the basis functions may be RBF kernels and the parameters $\boldsymbol{\theta}_n$ may include their location (center) and scale (width). Moreover, we assume a zero mean Gaussian prior for the weights \mathbf{w} , using a separate parameter α_n for the precision of each weight w_n , given by (3). Because we estimate different values for the parameters of each basis function, this model is very flexible and in order to avoid overfitting we use the sparsity enforcing prior of (9).

Unlike the typical linear model, which assumes that the basis functions are fixed and known in advance, we use the algorithm proposed in [1] to estimate the parameters $\boldsymbol{\theta}_n$ of the basis functions during model learning.

3.1 Adaptive Kernel Learning

The adaptive kernel learning algorithm [1] has been proposed to estimate the parameters of aRVM, which is based on an incremental learning algorithm for the typical RVM [4]. The incremental algorithm initially assumes an empty model, by setting $\alpha_i = \infty$, for all $i = 1, \dots, M$. Then, at each iteration one basis function may be either added to the model or re-estimated or removed from the current model.

Since we add only one basis function at each iteration, we maximize with respect to a *single parameter* α_i each time. Using the sparsity prior $p(\boldsymbol{\alpha})$ of (9) and keeping only the terms of the objective function that depend on α_i , the following function should be maximized with respect to α_i [14]

$$l_i^s = \frac{1}{2} \left(\log \alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2 + 2c\alpha_i}{\alpha_i + s_i} \right), \quad (17)$$

where

$$s_i = \boldsymbol{\phi}_i^T \mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i, \quad q_i = \boldsymbol{\phi}_i^T \mathbf{C}_{-i}^{-1} \hat{\mathbf{t}}, \quad (18)$$

$\phi_i = (\phi(\mathbf{x}_1; \boldsymbol{\theta}_i), \dots, \phi(\mathbf{x}_N; \boldsymbol{\theta}_i))^T$ and $\mathbf{C}_{-i} = \mathbf{B} + \sum_{j \neq i} \alpha_j \phi_j \phi_j^T$. In regression we have $\hat{\mathbf{t}} = \mathbf{t}$ and usually $\mathbf{B} = \beta \mathbf{I}$, while in classification \mathbf{B} and $\hat{\mathbf{t}}$ are given by (14) and (15) respectively. It has been shown in [1] that maximization of l_i^s with respect to α_i has a single maximum at:

$$\begin{aligned}\alpha_i &= \frac{s_i^2}{q_i^2 - (2c+1)s_i} && \text{if } q_i^2 > (2c+1)s_i, \\ \alpha_i &= \infty && \text{if } q_i^2 \leq (2c+1)s_i.\end{aligned}\quad (19)$$

Appropriate values for the basis function parameters θ_{ik} are determined using the quasi-Newton BFGS optimization method to maximize (17). The required derivatives can be analytically computed as:

$$\frac{\partial l_i^s}{\partial \theta_{ik}} = - \left(\frac{1}{\alpha_i + s_i} + \frac{q_i^2 + c\alpha_i}{(\alpha_i + s_i)^2} \right) r_i + \frac{q_i}{\alpha_i + s_i} w_i, \quad (20)$$

where

$$r_i \equiv \frac{1}{2} \frac{\partial s_i}{\partial \theta_{ik}} = \phi_i^T \mathbf{C}_{-i}^{-1} \frac{\partial \phi_i}{\partial \theta_{ik}}, \quad w_i \equiv \frac{\partial q_i}{\partial \theta_{ik}} = \mathbf{t}^T \mathbf{C}_{-i}^{-1} \frac{\partial \phi_i}{\partial \theta_{ik}}, \quad (21)$$

and $\phi_i = (\phi(\mathbf{x}_1; \boldsymbol{\theta}_i), \dots, \phi(\mathbf{x}_N; \boldsymbol{\theta}_i))^T$.

The adaptive kernel learning algorithm proceeds iteratively, selecting at each iteration the most appropriate basis function to add to the model.

3.2 Local Feature Selection

In (16) we assume that all basis function have the same parametric form, but different values $\boldsymbol{\theta}_n$ for the parameters. In order to facilitate feature selection we need to parameterize the kernel function so that it incorporates scaling factors for each dimension. Here, we consider anisotropic Gaussian kernel functions, which have a separate precision parameter h_{ni} for each feature i :

$$\phi(\mathbf{x}; \mathbf{m}_n, \mathbf{h}_n) = \exp \left[- \sum_{i=1}^d (h_{ni})^2 (x_i - m_{ni})^2 \right], \quad (22)$$

where $\mathbf{h}_n = (h_{n1}, \dots, h_{nd})^T$ and $\mathbf{m}_n = (m_{n1}, \dots, m_{nd})^T$. We call feature selection *local*, because we have assigned separate scaling factors for the features of each kernel. This approach is much more flexible to the typical feature selection approach, since different features are considered to be significant at different regions of the input space.

Notice that if we assign a very small value to a scaling factor h_{ni} of the n -th kernel, the corresponding feature x_i does not contribute to that kernel. Under the Bayesian framework, elimination of irrelevant features can be motivated by imposing a prior distribution on the scaling factors $\mathbf{h} = (\mathbf{h}_1^T, \dots, \mathbf{h}_M^T)^T$ that enforces sparsity. In analogy to the prior on the weights w_n , the prior distribution

that we use on \mathbf{h}_n is the Student's t distribution, which is known to give sparse solutions for few degrees of freedom [7]. Since a Student's t distributed random variable is equivalent to a Gaussian distributed random variable whose precision parameter is assumed Gamma distributed, we can write:

$$p(\mathbf{h}|\boldsymbol{\delta}) = \text{N}(\mathbf{h}|\mathbf{0}, \boldsymbol{\Delta}^{-1}), \quad (23)$$

with $\boldsymbol{\delta}_n = (\delta_{n1}, \dots, \delta_{nd})^T$, $\boldsymbol{\Delta} = \text{diag}\{\boldsymbol{\delta}_1, \dots, \boldsymbol{\delta}_M\}$ and

$$p(\boldsymbol{\delta}) = \prod_{n=1}^M \prod_{i=1}^d \text{Gamma}(\delta_{ni}|a, b), \quad (24)$$

In the above definition we set $a = b = 0$ to define an uninformative Gamma distribution.

Since we assume different parameter values h_{ni} for each kernel, the above model selects different features to contribute at each kernel. Therefore, this model is much more flexible to the typical feature selection approach that selects a set of significant features that is *common* for all the kernels. An example that demonstrates the advantages of the proposed model is presented in Section 4.

3.3 Parameter Learning

The learning method is similar to the adaptive kernel learning algorithm of [1]. It incrementally adds basis functions to an initially empty model and at the same time it assigns appropriate values to their parameters. Estimation of the parameters \mathbf{m}_n can be performed using the quasi-Newton BFGS method with derivatives given from (20).

Estimation of the scaling factors \mathbf{h} can also be performed using the adaptive kernel learning algorithm. However, we must take into account the existence of the prior distribution of (23) that we have assigned to the scaling factors. Therefore, we now want to jointly maximize the posterior $p(\boldsymbol{\alpha}, \mathbf{h}|\mathbf{t})$ with respect to the weight precisions $\boldsymbol{\alpha}$ and the scaling factors \mathbf{h} , which is given by

$$\log p(\boldsymbol{\alpha}, \mathbf{h}|\mathbf{t}) = \log p(\mathbf{t}|\boldsymbol{\alpha}, \mathbf{h}) + \log p(\mathbf{h}) + \log p(\boldsymbol{\alpha}) = l_i^s - \frac{1}{2} \mathbf{h}^T \boldsymbol{\Delta} \mathbf{h}. \quad (25)$$

This optimization is performed using (19) to update the weight precisions α_i and using a quasi-Newton optimization method to update the scaling factors h_{ni} . The required derivatives are obtained from (20) by adding the term corresponding to the derivative of the prior $p(\mathbf{h})$:

$$\frac{\partial \log p(\mathbf{h}|\mathbf{t})}{\partial h_{ni}} = \frac{\partial \log p(\mathbf{t}|\boldsymbol{\theta})}{\partial h_{ni}} - \delta_{ni} h_{ni}. \quad (26)$$

Furthermore, we also have to specify the update equations for the parameters $\boldsymbol{\delta}$ that define the precision of \mathbf{h} . By setting $a = b = 0$ in (24), we assume an

Algorithm 1. Feature Selection Using Adaptive Kernel Learning

1. Select an inactive basis function to add to the model (convert to active) as follows:
 - (a) Consider an initial set of inactive candidate basis functions by sampling their parameters at random.
 - (b) Optimize separately the parameters of each candidate basis function to maximize the marginal likelihood.
 - (c) Add to the model the candidate basis function that increases the marginal likelihood the most.
2. Optimize the parameters θ of all currently active basis functions.
3. Update hyperparameters α and noise precision β , using (19) and (12).
4. Update hyperparameters δ using (27).
5. Remove from the model any unnecessary active basis functions.
6. Repeat steps 1 to 5 until convergence.

uninformative prior distribution for them. Then, maximization of the likelihood with respect to δ gives

$$\delta_{ni} = \frac{1}{h_{ni}^2}. \quad (27)$$

In summary the proposed learning method, which is described in Algorithm 1, is based on the incremental adaptive kernel learning algorithm of [1], but it also takes into account the sparsity prior for the scaling factors h and updates the parameters δ that have been introduced to facilitate feature selection.

4 Numerical Experiments

The purpose of the first experiment is to demonstrate the feature selection capabilities of the proposed method. For this reason, we have generated samples from two two-dimensional zero-mean Gaussian distributions, each corresponding to one of the classes. More specifically, we selected the variance of the Gaussian distributions to be $s_1 = (1, 10)^T$ and $s_2 = (10, 1)^T$, so that only one feature is significant for discriminating each class. In Fig. 1 we show the estimated models using i) the RVM with adaptive kernel learning algorithm of [1] and ii) the proposed modification to incorporate feature selection. Notice, that the model obtained using the proposed approach contains only one basis function for each class, with scaling factors $\theta_1 = (0.6, 0.0)^T$ and $\theta_2 = (0.0, 0.4)^T$, therefore it successfully identifies the relevant features for each basis function. For this reason, it achieves better classification performance (measured by the percentage of misclassified examples), which is also shown in Fig. 1.

Furthermore, in order to evaluate the method we have performed experiments with several regression and classification datasets from the UCI Machine Learning Repository that were also used in [1]. More specifically, we estimate the generalization error of each method by performing ten-fold cross validation on each dataset. In regression, the error is the *mean square error*,

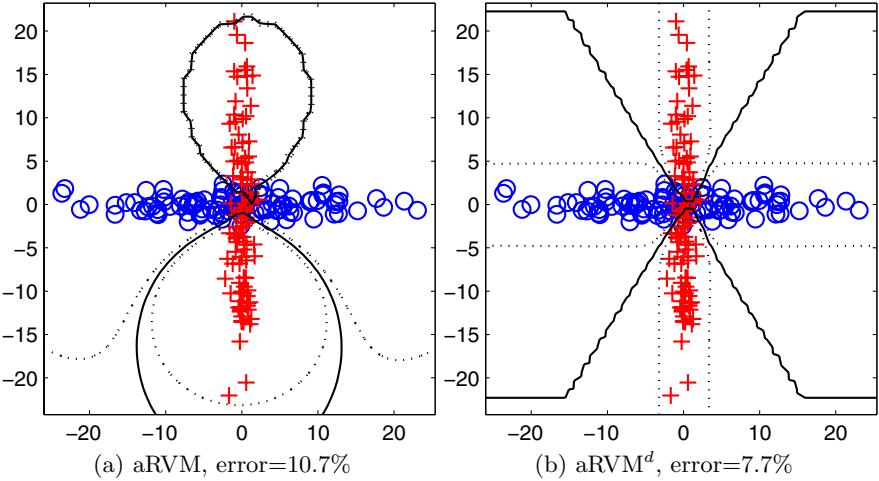


Fig. 1. Artificial classification example. Solution provided by aRVM classifiers: (a) without feature selection (b) with feature selection. Solid lines show the estimated decision boundary and dotted lines correspond to the points where the estimated probability of misclassification is 0.25.

$MSE = \sum_n (t_n - \hat{y}_n)^2 / N$, where t_n is the value given by the test set, \hat{y}_n the predicted value and N the number of test examples. In classification the error is the percentage of misclassified examples in the test set. We evaluate three methods; i) the typical RVM with Gaussian kernel (denoted as RVM), ii) adaptive RVM with learning of Gaussian kernel parameters Π (denoted as aRVM) and iii) the proposed adaptive RVM with simultaneous feature selection by learning the parameters of anisotropic Gaussian kernels (denoted as aRVM^d). The regression and classification results are shown in Table 1 and Table 2 respectively, where for each dataset the error and number of relevance vectors(RV) are presented. It can be observed that in most cases the proposed approach, which incorporates feature selection, provides improved performance compared to both the typical RVM model and the aRVM method Π .

Table 1. Experimental results on regression datasets

Dataset	patterns (N)	features (d)	RVM		aRVM		aRVM ^d	
			error	RVs	error	RVs	error	RVs
computer	209	6	30004	140.5	22379	5.0	4089	13.6
Boston	506	13	12.48	69.5	11.53	13.27	13.66	17.5
concrete	1030	8	44.204	140.2	34.515	9.10	28.868	42.3

Table 2. Experimental results on classification datasets

Dataset	patterns (N)	features (d)	RVM	aRVM	aRVM ^d
			error RVs	error RVs	error RVs
banana	5300	2	0.1092	12.1	0.1126
titanic	2200	3	0.2292	31.0	0.2270
image	2310	18	0.0390	34.6	0.0387
breast-cancer	277	9	0.2818	9.6	0.2844
pima	768	8	0.243	27.9	0.2303

5 Conclusions

In this work we have presented an approach to incorporate feature selection to the sparse Bayesian linear model by adapting the kernel learning approach of [1]. In contrast to typical feature selection approaches, the significance of each feature is assessed separately for each kernel. Therefore, for each kernel a different set of significant features is selected. This approach might be useful, for example in a classification problem when different features are significant for discriminating the examples of each class. Experimental results on several regression and classification datasets demonstrate the merits of the method. As a future work, we aim to test the effectiveness of this approach to bioinformatics problems (eg. gene microarray classification), where the number of features is very high and feature selection is a very important issue.

References

1. Tzikas, D., Likas, A., Galatsanos, N.: Sparse bayesian modeling with adaptive kernel learning. *IEEE Transactions on Neural Networks* (to appear)
2. Tipping, M.E.: Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 1, 211–244 (2001)
3. Schmolck, A., Everson, R.: Smooth relevance vector machine: a smoothness prior extension of the RVM. *Machine Learning* 68(2), 107–135 (2007)
4. Tipping, M.E., Faul, A.: Fast marginal likelihood maximisation for sparse Bayesian models. In: *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* (2003)
5. Krishnapuram, B., Hartemink, A.J., Figueiredo, M.A.T.: A Bayesian approach to joint feature selection and classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(9), 1105–1111 (2004)
6. Holmes, C.C., Denison, D.G.T.: Bayesian wavelet analysis with a model complexity prior. In: Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M. (eds.) *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*. Oxford University Press, Oxford (1999)
7. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, Heidelberg (2006)

MINLIP: Efficient Learning of Transformation Models*

Vanya Van Belle, Kristiaan Pelckmans, Johan A.K. Suykens, and Sabine Van Huffel

Katholieke Universiteit Leuven, ESAT-SCD
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
{vvanbell,kpelckma,johan.suykens,vanhuffe}@esat.kuleuven.be

Abstract. This paper studies a risk minimization approach to estimate a transformation model from noisy observations. It is argued that transformation models are a natural candidate to study ranking models and ordinal regression in a context of machine learning. We do implement a structural risk minimization strategy based on a Lipschitz smoothness condition of the transformation model. Then, it is shown how the estimate can be obtained efficiently by solving a convex quadratic program with $O(n)$ linear constraints and unknowns, with n the number of data points. A set of experiments do support these findings.

Keywords: Support vector machines, ranking models, ordinal regression.

1 Introduction

Non-linear methods based on ranking continue to challenge researchers in different scientific areas, see e.g. [57]. Problems of learning ranking functions come in different flavors, including ordinal regression, bipartite ranking and discounted ranking studied frequently in research on information retrieval. This problem will be considered in the context of Support Vector Machines (SVM) [11214] and convex optimization. We study the general problem where the output domain can be arbitrary (with possibly infinite members), but possess a natural ordering relation between the members. This general problem was studied before in [17], and results can be specified to the aforementioned specific settings by proper definition of the domain of the outputs (e.g. restricting its cardinality to $k < \infty$ or $k = 2$).

A main trend is the reduction of a ranking problem to a pairwise classification problem, bringing in all methodology from learning theory. It may however be argued that such an approach deflects attention from the real nature of the ranking problem. It is for example not clear that the complexity control (in a broad sense) which is successful for classification problems is also natural and efficient in the ranking setting. More specifically, it is often taken for granted that the measure of margin - successful in the setting of binary classification - has a natural counterpart in the ranking setting as the measure of *pairwise margin*, although it remains somewhat arbitrary how this is to be implemented exactly, see e.g. [4]. In order to approach such questions, we take an alternative

* KP is a postdoctoral researcher with FWO Flanders (A 4/5 SB 18605). S. Van Huffel is a full professor and J.A.K. Suykens is a professor at the Katholieke Universiteit Leuven, Belgium. This research is supported by GOA-AMBIORICS, CoE EF/05/006, FWO G.0407.02 and G.0302.07, IWT, IUAP P6/04, eTUMOUR (FP6-2002-LIFESCIHEALTH 503094).

approach: we will try to learn a single function $u : \mathbb{R}^d \rightarrow \mathbb{R}$, such that the natural order on \mathbb{R} induces the desired ranking (approximatively). Such a function is often referred to as a scoring, ranking, utility or health function depending on the context - we will use *utility function* in this text.

In the realizable case, techniques as complexity control, regularization or Occam's razor (in a broad sense) give a guideline to learn a specific function in case there are more functions exactly concordant with the observed data: a simpler function has a better chance of capturing the underlying relation. In short, we will argue that a utility function reproducing the observed order is less complex than another concordant function if the former is more smoothly related to the actual output values. That is, if there is an exact order relation between two variables, one can obviously find (geometrically) a monotonically increasing function between them. This argument relates ranking directly to what is well-studied in the statistical literature as transformation models, see e.g. [69]. Here the monotonically increasing mapping between utility function and output is referred to as the transformation function. Now, we define the complexity of a prediction rule for transformation models as being the Lipschitz constant of this transformation function. When implementing a risk minimization strategy based on these insights, the resulting methods are similar to the binary, hard margin SVMs, but do differ conceptually and computationally with existing ranking approaches. Also similar in spirit to the non-separable case in SVMs, it is indicated how slack variables can be used to relax the realizable case: we assume that an exactly concordant function can be found, were it not for incomplete observation of the patients' covariates.

This paper is organized as follows. Section 2 discusses in some detail the use of transformation models and its relation with ranking methods. Section 3 introduces an efficient estimator of such a transformation function, relying on ideas as thoroughly used in the machine learning literature. Section 4 gives insight how our estimator can be modified in the context of ordinal regression. Section 5 reports experimental results supporting the approach.

2 Transformation Models and Ranking Methods

In order to make the discussion more formal, we adopt the following notation. We work in a stochastic context, so we denote random variables and vectors as capital letters, e.g. X, Y, \dots , which follow an appropriate stochastic law P_X, P_Y, \dots , abbreviated (generically) as P . Deterministic quantities as constants and functions are represented in lower case letters (e.g. d, h, u, \dots). Matrices are denoted as boldface capital letters (e.g. $\mathbf{X}, \mathbf{D}, \dots$). Now we give a definition of a *transformation model*.

Definition 1 (Transformation Model). Let $h : \mathbb{R} \rightarrow \mathbb{R}$ be a strictly increasing function, and let $u : \mathbb{R}^d \rightarrow \mathbb{R}$ be a function of the covariates $X \in \mathbb{R}^d$. A Transformation Model (or TM) takes the following form

$$Y = h(u(X)). \quad (1)$$

Let ϵ be a random variable ('noise') independent of X , with cumulative distribution function $F_\epsilon(e) = P(\epsilon \leq e)$ for any $e \in \mathbb{R}$. Then a Noisy Transformation Model (NTM) takes the form

$$Y = h(u(X) + \epsilon). \quad (2)$$

Now the question reads as how to estimate the utility function $u : \mathbb{R}^d \rightarrow \mathbb{R}$ and the transformation model h from i.i.d. samples $\{(X_i, Y_i)\}_{i=1}^n$ without imposing any distributional (parametric) assumptions on the noise terms $\{\epsilon_i\}$.

Transformation models are often considered in the context of failure time models and survival analysis [8]. It should be noted that the approach which will be outlined sets the stage for deriving predictive models in this context. Note that in this context [3][6][9] one considers transformation models of the form $h^-(Y) = u(X) + \epsilon$, which are equivalent in case h is invertible, or $h^-(h(z)) = h(h^-(z)) = z$ for all z .

The relation with empirical risk minimization for ranking and ordinal regression goes as follows. The risk of a ranking function with respect to observations is often expressed in terms of Kendall's τ , Area Under The Curve or a related measure. Here we consider the (equivalent) measure of *disconcordance* (or one minus concordance) for a fixed function $u : \mathbb{R}^d \rightarrow \mathbb{R}$, where the probability concerns the two i.i.d. copies (X, Y) and (X', Y') :

$$\mathcal{C}(u) = P((u(X) - u(X'))(Y - Y') < 0). \quad (3)$$

Given a set of n i.i.d. observations $\{(X_i, Y_i)\}_{i=1}^n$,

$$\mathcal{C}_n(u) = \frac{2}{n(n-1)} \sum_{i < j} I((u(X_i) - u(X_j))(Y_i - Y_j) < 0), \quad (4)$$

where the indicator function $I(z)$ equals one if z holds, and equals zero otherwise. Empirical Risk Minimization (ERM) is then performed by solving

$$\hat{u} = \arg \min_{u \in \mathcal{U}} \mathcal{C}_n(u), \quad (5)$$

where $\mathcal{U} \subset \{u : \mathbb{R}^d \rightarrow \mathbb{R}\}$ is an appropriate subset of ranking functions, see e.g. [5] and citations. This approach however results in difficult and combinatorial optimization problems, and the current solution is to majorize the discontinuous indicator function with the Hinge loss, i.e. $I(z) \leq \max(0, 1 - z)$ yielding rankSVM [7]. The intrinsic problem with such an approach is that one has $O(n^2)$ number of constraints or unknowns in the final optimization problem, obstructing applicability (computationally) to many real life cases.

Now, there is an intrinsic relation with transformation models which circumvent such problems. The crucial observation here (again) is that *if a function $u : \mathbb{R}^d \rightarrow \mathbb{R}$ exists such that $\mathcal{C}_n(u) = 0$, one describes implicitly a monotonically increasing transformation function* (see Figure 1). In the case that $\mathcal{C}_n(u) = 0$ is not satisfied, we will adopt the noisy transformation model and use the error terms (slack variables) to model the deviance from this assumption. This reasoning is entirely similar as is used in formulating the hard margin Support Vector Machine, and its soft-margin variation.

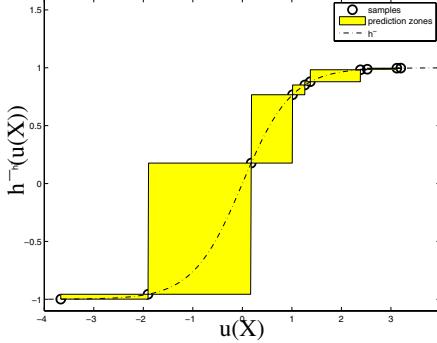


Fig. 1. The main observation relating ranking and transformation models is that if two variables $u(x)$ and y are perfectly *concordant*, they describe (implicitly) a monotonically increasing function $y = h(u(x))$. This means that a perfect ranking function corresponds with a (noiseless) transformation model. Moreover, if the samples are pairwise Lipschitz, there exists a Lipschitz transformation function. The yellow zones indicate possible function values on test samples.

3 MINLIP: A Convex Approach to Learning a Transformation Model

3.1 Lipschitz Smooth Functions and Transformation Models

In order to overcome the difficulties of implementing the estimator given in equation (5), we need one final ingredient. This concept will play a similar role as the margin in Support Vector Machines for classification. We will say that the univariate function h has a Lipschitz constant of $L \geq 0$ if $|h(z) - h(z')| \leq L|z - z'|$ for all $z, z' \in \mathbb{R}$, or equivalently

$$|h(u(x)) - h(u(x'))| \leq L|u(x) - u(x')|, \quad \forall x, x' \in \mathbb{R}^d. \quad (6)$$

Now, since h is monotonically increasing one has also $h(z) - h(z') \leq z - z'$ for all $z \geq z'$, and restricting attention to the samples $\{(x_i, y_i)\}_{i=1}^n$, one has the necessary and sufficient conditions $h(u(X_{(i)})) - h(u(X_{(i-1)})) \leq L(u(X_{(i)}) - u(X_{(i-1)}))$ for all $i = 2, \dots, n$. Here, we assume that the data obey a noiseless transformation model (as in (1)), and the samples are reindexed as $\{(X_{(i)}, Y_{(i)})\}_{i=1}^n$ where $Y_{(i-1)} \leq Y_{(i)}$ for all $i = 2, \dots, n$. Wrapping up results thus far gives us the following proposition:

Proposition 1 (Existence of h). *Given a set of samples $\{(X_{(i)}, Y_{(i)})\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$ and a function $u : \mathbb{R}^d \rightarrow \mathbb{R}$, such that $Y_{(i)} \geq Y_{(i-1)}$ for all $i = 2, \dots, n$, and*

$$Y_{(i)} - Y_{(i-1)} \leq L(u(X_{(i)}) - u(X_{(i-1)})), \quad \forall i = 2, \dots, n, \quad (7)$$

Then there exists a monotonically increasing function $h : \mathbb{R} \rightarrow \mathbb{R}$ such that the mapping x to y obeys $y = h(u(x))$ and h has Lipschitz constant L following (2) (see Figure 1).

Before using non-linear utility functions, we will consider only linear utilities in the next two sections.

3.2 Kernel Based Model

Since the function $u(x) = w^T x$ can be arbitrarily rescaled such that the corresponding transformation function has arbitrary Lipschitz constant (i.e. for any $c > 0$, one has $h(u(x)) = h'(u'(x))$ where $h'(z) = h(c^{-1}z)$ and $u'(x) = cu(x)$), we fix the norm $w^T w$ and try to find $u(x) = v^T x$ with $v^T v = 1$. Hence learning a transformation model with minimal Lipschitz constant of h can be written as

$$\min_{v, L} L^2 \text{ s.t. } \|v\|_2 = 1, Y_{(i)} - Y_{(i-1)} \leq L(v^T X_{(i)} - v^T X_{(i-1)}), \quad \forall i = 2, \dots, n \quad (8)$$

and equivalently substituting $w = Lv$ as

$$\min_w \frac{1}{2} w^T w \text{ s.t. } Y_{(i)} - Y_{(i-1)} \leq w^T X_{(i)} - w^T X_{(i-1)}, \quad \forall i = 2, \dots, n \quad (9)$$

which goes along similar lines as the hard margin SVM (see e.g. [11]). Remark that there is no need for an intercept term here. Observe that this problem has $n - 1$ linear constraints. We will refer to this estimator of w as MINLIP. We can rewrite this problem compactly as

$$\min_w \frac{1}{2} w^T w \text{ s.t. } \mathbf{D}\mathbf{X}w \geq \mathbf{D}\mathbf{Y}, \quad (10)$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a matrix with each row containing a sample, i.e. $\mathbf{X}_{(i)} = X_{(i)} \in \mathbb{R}^d$, $\mathbf{Y}_{(i)} = Y_{(i)} \in \mathbb{R}$. The matrix $\mathbf{D} \in \{-1, 0, 1\}^{(n-1) \times n}$ gives the first order differences of a vector, i.e. assuming no ties in the output, $\mathbf{D}_j \mathbf{Y} = Y_{(j+1)} - Y_{(j)}$ for all $j = 1, \dots, n-1$, with \mathbf{D}_j the j th row of \mathbf{D} . In the presence of ties $Y_{(j+1)}$ is replaced by $Y_{(i)}$, with i the smallest output value with $Y_{(i)} > Y_{(j)}$. Solving this problem as a convex QP can be done efficiently with standard mathematical solvers as implemented in MOSEK¹ or R-quadprog².

3.3 The Agnostic Case

The agnostic case deals with the case where one is not prepared to make the assumption that a function exists which will exactly extract in all cases the most relevant element. To model this, we impute a random variable ϵ with expected value zero, which acts additive on the contribution of the covariates (hence nonadditive on the final output for general function h). Hence our model becomes

$$Y = h(u(X) + \epsilon) = h(w^T X + \epsilon), \quad (11)$$

as in (2). Now we suggest how one can integrate the agnostic learning scheme with the Lipschitz-based complexity control. We will further specify the loss function $\ell : \mathbb{R} \rightarrow \mathbb{R}$ to the absolute value loss, or $\ell(\epsilon) = |\epsilon|$. The reason for doing so is threefold. At first, this loss function is known to be more robust to model misspecification and outliers (leverage points) than e.g. the squared loss $\ell(\epsilon) = \epsilon^2$. Secondly, this loss will result

¹ <http://www.mosek.org>

² <http://cran.r-project.org/web/packages/quadprog/index.html>

in sparse terms, i.e. many of the estimated error terms will be zero. This in turn can be exploited in order to obtain a compact representation of the estimate through the dual (as is the case for Support Vector Machines (SVMs) [14], and see the following subsection). Thirdly, the one-norm loss is found to perform well in the binary classification case as implemented in the SVMs. However, we stress that the choice of this loss is in some sense arbitrary, and should be tailored to the case study at hand. One can formalize the learning objective for a fixed value of $\gamma > 0$ with errors $\epsilon = (\epsilon_1, \dots, \epsilon_{n-1})^T \in \mathbb{R}^{n-1}$:

$$\min_{w, \epsilon} \frac{1}{2} w^T w + \gamma \|\epsilon\|_1 \quad \text{s.t. } \mathbf{D}(\mathbf{X}w + \epsilon) \geq \mathbf{DY}, \quad (12)$$

where $\|\epsilon\|_1 = \sum_{i=1}^n |\epsilon_i|$. This problem can again be solved as a convex quadratic program.

3.4 A Nonlinear Extension Using Mercer Kernels

Consider the model

$$u(x) = w^T \varphi(x), \quad (13)$$

where $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^{d_\varphi}$ is a mapping of the data to a high dimensional feature space (of dimension d_φ , possibly infinite). Now $w \in \mathbb{R}^{d_\varphi}$ is a (possibly) infinite dimensional vector of unknowns. Let $\Phi = [\varphi(X_{(1)}), \dots, \varphi(X_{(n)})]^T \in \mathbb{R}^{n \times d_\varphi}$. Then we can write the learning problem concisely as

$$\min_w \frac{1}{2} w^T w \quad \text{s.t. } \mathbf{D}\Phi w \geq \mathbf{DY}, \quad (14)$$

with the matrix \mathbf{D} defined as before. This problem can be solved efficiently as a convex Quadratic Programming (QP) problem. The Lagrange dual problem becomes

$$\min_{\alpha} \frac{1}{2} \alpha^T \mathbf{DKD}^T \alpha - \alpha^T \mathbf{DY} \quad \text{s.t. } \alpha \geq 0_{n-1} \quad (15)$$

where the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ contains the kernel evaluations such that $\mathbf{K}_{ij} = \varphi(X_i)^T \varphi(X_j)$ for all $i, j = 1, \dots, n$. The estimated \hat{u} can be evaluated at any point $x \in \mathbb{R}^d$ as

$$\hat{u}(x) = \hat{\alpha}^T \mathbf{DK}_n(x), \quad (16)$$

where $\hat{\alpha}$ solves (15), and $K_n(x) = (K(X_1, x), \dots, K(X_n, x))^T \in \mathbb{R}^n$. A similar argument gives the dual of the agnostic learning machine of Subsection 3.3 (12), see e.g. [11][12][14]:

$$\min_{\alpha} \frac{1}{2} \alpha^T \mathbf{DKD}^T \alpha - \alpha^T \mathbf{DY} \quad \text{s.t. } \begin{cases} -\gamma 1_n \leq \mathbf{D}^T \alpha \leq \gamma 1_n \\ \alpha \geq 0_{n-1}, \end{cases} \quad (17)$$

with \mathbf{K} as above and the resulting estimate can be evaluated as in (16) without computing explicitly \hat{u} . It is seen that the nonlinear model can be estimated using a pre-defined kernel function, and without explicitly defining the mapping $\varphi(\cdot)$.

4 Learning for Ordinal Regression

Consider now the situation where the output takes a finite number of values - say $k \in \mathbb{N}$ - and where the k different classes possess a natural ordering relation. Instead of ranking all samples with its closest sample, one has to enumerate the rankings of all samples with certain output levels with all samples possessing the closest non-equal output level. However, when only observing a constant number k different output levels, this procedure can increase the number of constraints in the estimation problem to $\mathcal{O}(n^2)$. To cope with this issue, we introduce unknown thresholds $\{v_j\}_{j=1}^{k-1}$ on the utility function, corresponding with known output levels $z_j = Y^j + \frac{1}{2}(Y^{j+1} - Y^j)$. This implies that one has to compare each sample only twice, namely with thresholds z_j and z_{j+1} for each data point in class j . This problem can be formulated as

$$\min_{\overline{w}, \epsilon} \frac{1}{2} w^T w + \gamma \|\epsilon\|_1 \quad \text{s.t.} \quad \begin{cases} \mathbf{D}(\overline{\Phi}\overline{w} + \epsilon) \geq \mathbf{D}\overline{\mathbf{Y}}, \\ v_j \geq v_{j-1}, \forall j = 2, \dots, k-1, \end{cases} \quad (18)$$

with

$$\overline{w} = \begin{bmatrix} w \\ v \end{bmatrix} \quad \overline{\Phi} = \begin{bmatrix} \Phi & 0 \\ 0 & I \end{bmatrix} \quad \overline{\mathbf{Y}} = \begin{bmatrix} \mathbf{Y} \\ z \end{bmatrix}, \quad (19)$$

where \mathbf{D} needs to be build in such a way that $\mathbf{D}\overline{\Phi}\overline{w}$ equals the difference between the utility of each point and the utility of the nearest threshold.

5 Application Studies

5.1 Ordinal Regression

In a first example 6 regression datasets³ are used to compare the performance of the minlip model with two methods described in [4] (see Table 1). Both of these methods optimize multiple thresholds to define parallel discriminant hyperplanes for the ordinal levels. The first method (EXC) explicitly imposes the ordering of the thresholds, whereas this is done implicitly in the second method (IMC). Tuning of the Gaussian kernel parameter and the regularization parameter was performed with 10-fold cross-validation on an exponential grid. After an initial search, a finer search was performed in the neighborhood of the initial optimum. The datasets are divided into 20 folds with 10 equal-frequency bins, as in [4]. The generalization performance of the minlip method is clearly better than for the other methods. The IMC method performs best on the small dataset, but the minlip performance is better on larger datasets. Remark that the results on EXC and IMC obtained here are better than reported in [4].

In a second experiment, the performance and calculation time of the minlip model and standard rankSVM are compared on the pyrimidines dataset. Figure 2 shows the concordance, mean average error and calculation time when varying the number of training data points from 5 to 50. The concordance and error of both methods are comparable but for an increasing number of training data points the calculation time is considerably higher for the rankSVM method.

³ These regression datasets are available at

<http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html>

Table 1. Test results of minlip, EXC and IMC using a Gaussian kernel. The targets of the datasets were discretized by 10 equal-frequency bins. The results are averaged over 20 trials.

dataset	mean zero-one error			mean absolute error		
	minlip	EXC	IMC	minlip	EXC	IMC
pyrimidines	0.65 ± 0.09	0.70 ± 0.09	0.62 ± 0.07	1.01 ± 0.16	1.22 ± 0.22	1.00 ± 0.12
triazines	0.66 ± 0.06	0.72 ± 0.00	0.71 ± 0.02	1.19 ± 0.12	1.34 ± 0.00	1.27 ± 0.07
wisconsin	0.91 ± 0.03	0.89 ± 0.03	0.88 ± 0.03	2.33 ± 0.11	2.30 ± 0.17	2.25 ± 0.13
machine CPU	0.36 ± 0.04	0.55 ± 0.06	0.42 ± 0.09	0.54 ± 0.09	0.77 ± 0.07	0.69 ± 0.11
auto MPG	0.49 ± 0.04	0.55 ± 0.02	0.55 ± 0.03	0.62 ± 0.14	0.76 ± 0.05	0.75 ± 0.06
Boston housing	0.44 ± 0.04	0.50 ± 0.03	0.48 ± 0.03	0.54 ± 0.08	0.71 ± 0.06	0.63 ± 0.05

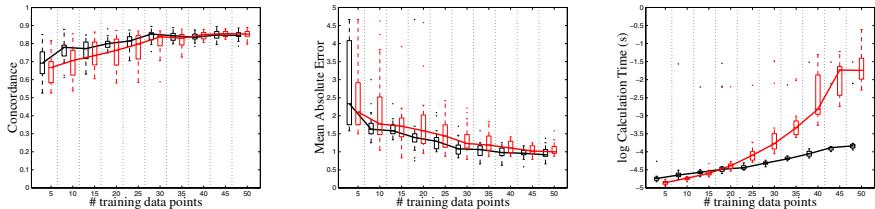


Fig. 2. Comparison between minlip (black) and the standard rankSVM (grey) on the pyrimidines dataset. The performance (concordance and mean absolute error are illustrated) of both methods is comparable, but for a reasonable number of training points, the calculation time is considerably lower for the first method.

5.2 Movie Recommendations

Our last application is a movie-recommendation task⁴. The data consists of the scores for 6040 viewers on 3952 movies. The goal is to predict the scoring of user i on movie j . We use the scorings of 1000 viewers as covariates to predict the scoring of the other viewers as follows

$$\hat{s}_{i,k} = \sum_{j=1}^{1000} w_{i,j} s_{j,k},$$

where $\hat{s}_{i,k}$ indicates the predicted score of user i on movie k , $w_{i,j}$ is the weight or "importance" of user j to predict the score given by user i . $s_{j,k}$ represents the score of movie k given by user j . The 1000 viewers with the highest number of rated movies were selected as reference viewers. Another 1000 (random) viewers were used as a validation set to tune the regularization parameter and the imputation value for scores in case a reference viewer did not score a certain movie. The values for the regularization parameter were selected after 10-fold cross-validation on an exponential grid. We chose two possible values for the imputation parameter: 3, which is the mean of all possible scores, and 2, which is one score lower than the previous one, indicating that the reason for not seeing a movie could be that one is not interested in the movie. For the 4040 remaining viewers, the first half of the rated movies were used for training, the second

⁴ Data available on <http://www.grouplens.org/node/73>

half for testing. The performance of the minlip method was compared with 3 other methods:

- **linear regression (LREG):** The score of the new user is found as a linear combination of the scores of the 1000 reference users.
- **nearest neighbor classification (NN):** This method searches the reference viewer for whom the scores are most similar to the scores of the new user. The score of the most similar reference viewer is considered as predicted score for the new viewer.
- **vector similarity (VSIM):** This algorithm [2] is based on the notion of similarity between two datapoints. The correlation between the new user and the reference users are used as weights $w_{k,i}$ in the formula: $\hat{s}_{k,j} = \bar{s}_k + a \sum_{i=1}^n w_{k,i}(s_{i,j} - \bar{s}_i)$, where \bar{s}_i represents the mean score for viewer i and a is a normalization constant such that $\sum_i |w_{k,i}| = 1$.

Three different performance measure were used for comparison of the methods:

- **mean zero-one error (MZOE)**
- **mean absolute error (MAE)**
- **concordance (CONC):** measuring the concordance of the test set within the training set, defined as:

$$\text{CONC}_n(u) = \frac{\sum_{i=1}^{n_t} \sum_{j=1}^n I[(u(X_j) - u(X_i))(T_j - T_i) > 0]}{n_t n},$$
with n and n_t the number of datapoints in the training and test set respectively.

Figure 3 compares all 4 methods for the 3 considered performance measures. The mean zero-one and mean absolute error should be as small as possible, while the concordance should be close as large as possible. The LREG method performs the least on all measures. The VSIM method results in a good average precision and low error measures, whereas the NN method is better in obtaining a high concordance. The advantage of the minlip method is that it performs good on all the measures.

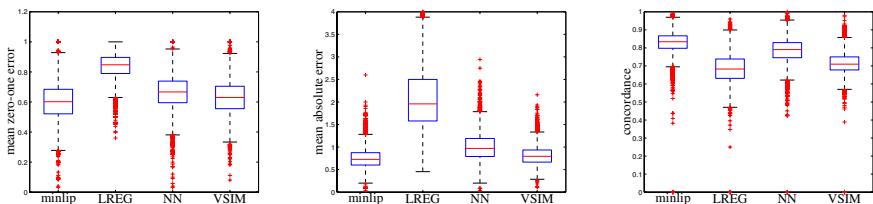


Fig. 3. Performance comparison of 4 methods: minlip (linear kernel), linear regression (LREG), nearest neighbor (NN) and vector similarity (VSIM). Three different performance measure were used. LREG performs the least on all measures. VSIM has low errors, whereas the NN method has a high concordance. The advantage of the minlip method is that it performs well on all the investigated performance measures.

6 Conclusions

This paper proposed an efficient estimator of a transformation model from noisy observations. The motivation for considering this problem is given by describing its relation to (i) the problem of learning ranking functions, and (ii) its relevance to estimating statistical models e.g. in a context of survival analysis. The latter topic will be the focus of subsequent work. We conducted two experiments to illustrate the use of this estimator: a first example on the prediction of the rankings of movies showed a good performance on different measures where other methods performed worse regarding at least one measure. In a second example on ordinal regression, we illustrate the reduction in calculation time in comparison with the standard rankSVM method, without reduction in performance.

References

1. Agarwal, S., Graepel, T., Herbrich, R., Har-Peled, S., Roth, D.: Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research* 6, 393–425 (2005)
2. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52 (1998)
3. Cheng, S.C., Wei, L.J., Ying, Z.: Predicting Survival Probabilities with Semiparametric Transformation Models. *Journal of the American Statistical Association* 92(437), 227–235 (1997)
4. Chu, W., Keerthi, S.S.: New approaches to support vector ordinal regression. In: *ICML*, pp. 145–152 (2005)
5. Clémençon, S., Lugosi, G., Vayatis, N.: Ranking and Scoring Using Empirical Risk Minimization. In: Auer, P., Meir, R. (eds.) *COLT 2005. LNCS (LNAI)*, vol. 3559, pp. 1–15. Springer, Heidelberg (2005)
6. Dabrowska, D.M., Doksum, K.A.: Partial likelihood in transformation models with censored data. *Scandinavian Journal of Statistics* 15(1), 1–23 (1988)
7. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. In: *Advances in Large Margin Classifiers*, pp. 115–132. MIT Press, Cambridge (2000)
8. Kalbfleisch, J.D., Prentice, R.L.: *The Statistical Analysis of Failure Time Data*. Wiley series in probability and statistics. Wiley, Chichester (2002)
9. Koenker, R., Geling, O.: Reappraising Medfly Longevity: A Quantile Regression Survival Analysis. *Journal of the American Statistical Association* 96(454), 458–468 (2001)
10. Pelckmans, K., Suykens, J.A.K., De Moor, B.: A Risk Minimization Principle for a Class of Parzen Estimators. In: *Advances in Neural Information Processing Systems 20*, pp. 1–8. MIT Press, Cambridge (2008)
11. Shawe-Taylor, J., Cristianini, N.: *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge (2004)
12. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: *Least Squares Support Vector Machines*. World Scientific, Singapore (2002)
13. Van Belle, V., Pelckmans, K., Suykens, J.A.K., Vanhuffel, S.: Support Vector Machines for Survival Analysis. In: *Proceedings of the Third International Conference on Computational Intelligence in Medicine and Healthcare, CIMED*, Plymouth, UK, July 25–27, pp. 1–6 (2007)
14. Vapnik, V.N.: *Statistical Learning Theory*. Wiley and Sons, Chichester (1998)

Efficient Uncertainty Propagation for Reinforcement Learning with Limited Data

Alexander Hans^{1,2} and Steffen Udluft^{1,*}

¹ Siemens AG, Corporate Technology, Information & Communications,
Learning Systems, Otto-Hahn-Ring 6, D-81739 Munich, Germany

{alexander.hans.ext,steffen.udluft}@siemens.com

² Ilmenau Technical University, Neuroinformatics and Cognitive Robotics Lab,
P.O. Box 100565, D-98684 Ilmenau, Germany

Abstract. In a typical reinforcement learning (RL) setting details of the environment are not given explicitly but have to be estimated from observations. Most RL approaches only optimize the expected value. However, if the number of observations is limited considering expected values only can lead to false conclusions. Instead, it is crucial to also account for the estimator's uncertainties. In this paper, we present a method to incorporate those uncertainties and propagate them to the conclusions. By being only approximate, the method is computationally feasible. Furthermore, we describe a Bayesian approach to design the estimators. Our experiments show that the method considerably increases the robustness of the derived policies compared to the standard approach.

Keywords: Reinforcement learning, model-based, uncertainty, Bayesian modeling.

1 Introduction

In reinforcement learning (RL) [12] one is concerned with finding a policy, i.e., a mapping from states to actions, that moves an agent optimally in an environment assumed to be a Markov decision process (MDP) $M := (S, A, P, R)$ with a state space S , a set of possible actions A , the system dynamics, defined as probability distribution $P : S \times A \times S \rightarrow [0, 1]$, which gives the probability of reaching state s' by executing action a in state s , and a reward function $R : S \times A \times S \rightarrow \mathbb{R}$, which determines the reward for a given transition. If the parameters of the MDP are known a priori, an optimal policy can be determined, e.g., using dynamic programming. Often, however, the MDP's parameters are not known in advance. A common way of handling this situation is model-based RL, where one first estimates a model of the MDP from a number of observations and then finds an optimal policy w.r.t. that model. In general, such a policy will not be optimal w.r.t. the real MDP. Especially in case of a limited number of observations the estimated MDP has a high probability to differ from the real one substantially.

* The authors would like to thank the reviewers for very useful comments.

In this case, it is in particular possible to derive a policy that will perform badly when applied to the real MDP.

By incorporating the model estimators' uncertainties into the determination of the policy it is possible to weaken this problem. In recent work by Schneegass et al. [10] uncertainty propagation (UP) was applied to the Bellman iteration to determine the Q-function's [12] uncertainty and derive uncertainty incorporating policies. While the algorithm described in [10] provides significant advantages over methods not considering uncertainty, it adds a huge computational burden for updating the covariance matrix in each iteration. In this paper, we propose an algorithm called the *diagonal approximation of uncertainty incorporating policy iteration* (DUIPI) for discrete MDPs that represents an efficient way of using UP to incorporate the model's uncertainty into the derived policy by only considering the diagonal of the covariance matrix. Only considering the diagonal neglects the correlations between the state-action pairs, which in fact are small for many RL problems, where on average different state-action pairs share only little probabilities to reach the same successor states. DUIPI is easier to implement and, most importantly, lies in the same complexity class as the standard Bellman iteration and is therefore computationally much cheaper than the method considering the full covariance matrix. Although some of the results obtained with DUIPI are not as good as those of the full-matrix method, the robustness of the resulting policies is increased considerably, compared to the standard Bellman iteration, which does not regard the uncertainty. In this context it furthermore is advisable to use Bayesian statistics to model the *a posteriori* distributions of the transition probabilities and rewards in order to access the estimators' uncertainties properly. Additionally, it allows the specification of prior knowledge and the user's belief.

There have already been a number of contributions that consider uncertainties when estimating MDPs. E.g., the framework of robust MDPs has widely been studied (e.g., [8][1]), in which one assumes that all uncertainties can only lie within a bounded set. One tries to find policies optimizing the worst case within that set, which often results in too conservative policies. Within the context of Bayesian RL, incorporation of prior knowledge about confidence and uncertainty directly into the approached policy is possible. E.g., Engel et al. applied Gaussian processes for policy evaluation by updating a prior distribution over value functions to posteriors by observing samples from the MDP [4][5]. Ghavamzadeh and Engel presented additional Bayesian approaches to model-free RL [6][7]. Using Gaussian processes inherently introduces a measure of uncertainty based on the number of samples. When dealing with model-based approaches, however, one starts with a natural local measure of the uncertainty of the transition probabilities and the rewards. In that context, related to the present paper is work by Delage and Mannor [3], who used convex optimization to solve the percentile problem and applied it to the exploration-exploitation trade-off. Model-based interval estimation (MBIE) was also used for efficient exploration by using local uncertainty to derive optimistic exploration policies, e.g., [11].

The remainder of the paper is organized as follows. In sec. 2 we describe how to incorporate knowledge of uncertainty into the Bellman iteration using UP, sec. 3 presents ways of parameter estimation. Experiments and results are presented in sec. 4. Sec. 5 finishes the paper with a short conclusion.

2 Incorporation of Uncertainty

Our notion of uncertainty is concerned with the uncertainty that stems from the ignorance of the exact properties of the real MDP, as they are usually unknown and must be estimated from observations. With an increasing number of observations the uncertainty decreases; in the limit of an infinite number of observations of every possible transition the uncertainty vanishes as the true properties of the MDP are revealed. For a given number of observations the uncertainty depends on the inherent stochasticity of the MDP; if the MDP is known to be completely deterministic, one observation of a transition is sufficient to determine all properties of that transition; the more the MDP is stochastic, the more uncertainty will remain for a fixed number of observations. It is important to distinguish this uncertainty from an MDP's inherent stochasticity.

We want to use the knowledge of uncertainty to determine an optimal Q-function Q^* with its uncertainty σQ^* . In a second step it is then possible to change the Bellman iteration to not only regard a Q-value but also its uncertainty, resulting in a policy that generally prefers actions that have a low probability of leading to an inferior long-term reward.

2.1 Determining the Q-Function's Uncertainty

To obtain the Q-function's uncertainty, we use the concept of uncertainty propagation (UP), also known as Gaussian error propagation (e.g., [2]), to propagate the uncertainties of the measurements, i.e., the transition probabilities and the rewards, to the conclusions, i.e., the Q-function and policy. The uncertainty of values $f(x)$ with $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is determined as $(\sigma f)^2 = \sum_i \left(\frac{\partial f}{\partial x_i} \right)^2 (\sigma x_i)^2$. The update step of the Bellman iteration,

$$Q^m(s, a) := \sum_{s'} \hat{P}(s'|s, a) \left[\hat{R}(s, a, s') + \gamma V^{m-1}(s') \right], \quad (1)$$

can be regarded as a function of the estimated transition probabilities \hat{P} and rewards \hat{R} , and the Q-function of the previous iteration Q^{m-1} (V^{m-1} is a subset of Q^{m-1}), that yields the updated Q-function Q^m . Applying UP to the Bellman iteration, one obtains an update equation for the Q-function's uncertainty:

$$\begin{aligned} (\sigma Q^m(s, a))^2 &:= \sum_{s'} (D_{QQ})^2 (\sigma V^{m-1}(s'))^2 + \sum_{s'} (D_{QP})^2 (\sigma \hat{P}(s'|s, a))^2 + \\ &\quad \sum_{s'} (D_{QR})^2 (\sigma \hat{R}(s, a, s'))^2, \end{aligned} \quad (2)$$

$$D_{QQ} = \gamma \hat{P}(s'|s, a), \quad D_{QP} = \hat{R}(s, a, s') + \gamma V^{m-1}(s'), \quad D_{QR} = \hat{P}(s'|s, a).$$

V^m and σV^m have to be set depending on the desired type of the policy (stochastic or deterministic) and whether policy evaluation or policy iteration is performed. E.g., for policy evaluation of a stochastic policy π

$$V^m(s) = \sum_a \pi(a|s) Q^m(s, a), \quad (3)$$

$$(\sigma V^m(s))^2 = \sum_a \pi(a|s)^2 (\sigma Q^m(s, a))^2. \quad (4)$$

For policy iteration, according to the Bellman optimality equation and resulting in the Q-function Q^* of an optimal policy, $V^m(s) = \max_a Q^m(s, a)$ and $(\sigma V^m(s))^2 = (\sigma Q^m(s, \arg \max_a Q^m(s, a)))^2$.

Using the estimators \hat{P} and \hat{R} with their uncertainties $\sigma \hat{P}$ and $\sigma \hat{R}$ and starting with an initial Q-function Q^0 and corresponding uncertainty σQ^0 , e.g., $Q^0 := 0$ and $\sigma Q^0 := 0$, through the update equations (1) and (2) the Q-function and corresponding uncertainty are updated in each iteration and converge to Q^π and σQ^π for policy evaluation and Q^* and σQ^* for policy iteration. Q^* and σQ^* can be used to obtain the function

$$Q_u(s, a) = Q^*(s, a) - \xi \sigma Q^*(s, a), \quad (5)$$

specifying a performance limit which, when the policy π^* is applied to the real MDP, will be exceeded with probability $\Pr(Z(s, a) > Q_u(s, a)) = F(\xi)$, where Z is the (unknown) Q-function of π^* for the real MDP. $F(\xi)$ depends on the distribution class of Q . E.g., if Q is normally distributed, F is the distribution function of the standard normal distribution. Note that a policy based on Q_u , i.e., $\pi_u(s) = \arg \max_a Q_u(s, a)$, does not in general improve the performance limit, as Q_u considers the uncertainty only for one step. In general, Q_u does not represent π_u 's Q-function, posing an inconsistency. To use the knowledge of uncertainty for maximizing the performance limit (as opposed to the expectation), the uncertainty needs to be incorporated into the policy-improvement step.

2.2 Uncertainty-Aware Policy Iteration

The policy-improvement step is contained within the Bellman optimality equation as $\max_a Q^m(s, a)$. Alternatively, determining the optimal policy in each iteration as

$$\forall s : \pi^m(s) := \arg \max_a Q^m(s, a) \quad (6)$$

and then updating the Q-function using this policy, i.e.,

$$\forall s, a : Q^m(s, a) := \sum_{s'} \hat{P}(s'|s, a) \left[\hat{R}(s, a, s') + \gamma Q^{m-1}(s', \pi^{m-1}(s)) \right], \quad (7)$$

yields the same solution. To determine a so-called *certain-* or ξ -optimal policy that maximizes the performance limit for a given ξ , the update of the policy

must not choose the optimal action w.r.t. to the maximum over the Q-values of a particular state but the maximum over the Q-values minus their weighted uncertainty:

$$\forall s : \pi^m(s) := \arg \max_a [Q^m(s, a) - \xi \sigma Q^m(s, a)]. \quad (8)$$

In each iteration, the uncertainty σQ^m has to be updated as described in sec. 2.1 setting V^m and σV^m as for deterministic policy evaluation.

The parameter ξ controls the influence of the uncertainty on the policy. Choosing a positive ξ yields uncertainty avoiding policies, with increasing ξ a worst-case optimal policy is approached. A negative ξ results in uncertainty seeking behavior.

2.3 Non-convergence of DUIPI for Deterministic Policy Iteration

While it has been shown that conventional policy iteration in the framework of MDPs is guaranteed to converge to a deterministic policy [9], for ξ -optimal policies derived by the algorithm presented in sec. 2.2 this is not necessarily the case. When considering a Q-value's uncertainty for action selection, there are two effects that contribute to an oscillation of the policy and consequently non-convergence of the corresponding Q-function.

First, there is the effect mentioned in [10] of a bias on $\xi \sigma Q(s, \pi(s))$ being larger than $\xi \sigma Q(s, a), a \neq \pi(s)$, if π is the evaluated policy and $\xi > 0$. DUIPI is not affected by this problem due to the ignorance of covariances between Q and R . Second, there is another effect (by which DUIPI is affected) causing an oscillation when there is a certain constellation of Q-values and corresponding uncertainties of concurring actions. Consider two actions a_1 and a_2 in a state s with similar Q-values but different uncertainties, a_1 having an only slightly higher Q-value but a larger uncertainty. The uncertainty-aware policy improvement step would alter π^m to choose a_2 , the action with the smaller uncertainty. However, the fact that this action is inferior might only become obvious in the next iteration when the value function is updated for the altered π^m (and now implying the choice of a_2 in s). In the following policy improvement step the policy will be changed back to choose a_1 in s , since now the Q-function reflects the inferiority of a_2 . After the next update of the Q-function, the values for both actions will be similar again, because now the value function implies the choice of a_1 and the bad effect of a_2 affects $Q(s, a_2)$ only once.

2.4 Risk-Reduction by Diversification through Stochastic Policies

With stochastic policies it is possible to construct an update-scheme that is guaranteed to converge, thus solving the problem of non-convergence. Moreover, it is intuitively clear that for $\xi > 0$ ξ -optimal policies should be stochastic as one tries to decrease the risk of obtaining a low long-term reward (because the wrong MDP has been estimated) by diversification.

The resulting algorithm initializes the policy with equiprobable actions. In each iteration, the probability of the best action according to Q_u^m (equation (5)) is increased by $1/m$, m being the current iteration, while the probabilities of all other actions are decreased accordingly:

$$\forall s, a : \pi^m(a|s) := \begin{cases} \min(\pi^{m-1}(a|s) + 1/m, 1), & \text{if } a = a_{Q_u^{m-1}}(s) \\ \frac{\max(1 - \pi(s, a_{Q_u^{m-1}}(s)) - 1/m, 0)}{1 - \pi(s, a_{Q_u^{m-1}}(s))} \pi^{m-1}(a|s), & \text{otherwise} \end{cases} \quad (9)$$

$a_{Q_u^{m-1}}(s)$ denotes the best action according to Q_u^{m-1} , i.e., $a_{Q_u^{m-1}}(s) = \arg \max_a Q^{m-1}(s, a) - \xi \sigma Q^{m-1}(s, a)$. Due to the harmonically decreasing change rate convergence as well as reachability of all possible policies are ensured.

3 Modeling of Estimators and Their Uncertainty

There are several ways of modeling the estimators for the transition probabilities P and the reward R . In the following we will present the frequentist approach using relative frequency as well as a Bayesian approach.

3.1 Frequentist Estimation

In the frequentist paradigm the relative frequency is used as the expected transition probability. The uncertainty of the according multinomial distribution is assumed to be

$$(\sigma \hat{P}(s'|s, a))^2 = \frac{\hat{P}(s'|s, a)(1 - \hat{P}(s'|s, a))}{n_{sa} - 1}, \quad (10)$$

where n_{sa} denotes the number of observed transitions from (s, a) .

Using the same concept for the rewards and assuming a normal distribution, the mean of all observed rewards of a transition (s, a, s') is used as reward expectation, their uncertainties are

$$(\sigma \hat{R}(s, a, s'))^2 = \frac{\text{var}(\hat{R}(s, a, s'))}{n_{sas'} - 1}, \quad (11)$$

with $n_{sas'}$ being the number of observed transitions (s, a, s') .

Although the estimation of the transition probabilities using relative frequency usually leads to good results in practice, the corresponding uncertainty estimation is problematic if there are only a few observations, because in that case the uncertainties are often underestimated. For instance, if a specific transition is observed twice out of two tries ($n_{sas'} = n_{sa} = 2$), its uncertainty $\sigma \hat{P}(s'|s, a) = 0$.

3.2 Bayesian Estimation

Assuming all transitions from different state-action pairs to be independent of each other and the rewards, the transitions can be modeled as multinomial distributions. In a Bayesian setting, where one assumes a prior distribution over the

parameter space $P(s_k|s_i, a_j)$ for given i and j , the Dirichlet distribution with density

$$\Pr(P(s_1|s_i, a_j), \dots, P(s_{|S|}|s_i, a_j))_{\alpha_{ij1}, \dots, \alpha_{ij|S|}} = \frac{\Gamma(\alpha_{ij})}{\prod_{k=1}^{|S|} \Gamma(\alpha_{ijk})} \prod_{k=1}^{|S|} P(s_k|s_i, a_j)^{\alpha_{ijk}-1}, \quad (12)$$

$\alpha_{ij} = \sum_{k=1}^{|S|} \alpha_{ijk}$, is a conjugate prior with posterior parameters $\alpha_{ijk}^d = \alpha_{ijk} + n_{s_i a_j s_k}$, $\alpha_{ij}^d = \sum_{k=1}^{|S|} \alpha_{ijk}^d$. Choosing the expectation of the posterior distribution as the estimator, i.e., $\hat{P}(s_k|s_i, a_j) = \alpha_{ijk}^d / \alpha_{ij}^d$, the uncertainty of \hat{P} is

$$(\sigma \hat{P}(s_k|s_i, s_j))^2 = \frac{\alpha_{ijk}^d (\alpha_{ij}^d - \alpha_{ijk}^d)}{(\alpha_{ij}^d)^2 (\alpha_{ij}^d + 1)}. \quad (13)$$

Note that $\alpha_i = 0$ results in a prior that leads to the same estimates and slightly lower uncertainties compared to the frequentist modeling of sec. 3.1. On the other hand, setting $\alpha_i = 1$ leads to a flat, maximum entropy prior that assumes all transitions from a state to all other states equally probable.

Both settings, $\alpha_i = 0$ and $\alpha_i = 1$, represent extremes that we believe are unreasonable for most applications. Instead, we model our prior belief by setting $\alpha_i = \frac{m}{|S|}$, where m is the average number of expected successor states of all state-action pairs and $|S|$ is the total number of states. This choice of α_i realizes an approximation of a maximum entropy prior over a subset of the state space with a size of m states. This way most of the probability is “distributed” among any subset of m states that have actually been observed, the probability of all other (not observed) successor states becomes very low. Compared to the maximum entropy prior with $\alpha_i = 1$ one needs only a few observations for the actually observed successor states to be much more probable than not observed ones. At the same time, the estimation of the uncertainty is not as extreme as the frequentist one, since having made the same observation twice does not cause the uncertainty to become zero. Estimating m from the observations can easily be added.

4 Experiments

We conducted experiments with DUIPI as presented here and the full-matrix algorithm using the frequentist as well as the Bayesian estimators described in the previous section¹.

4.1 Benchmark: Wet-Chicken 2D

The benchmark problem used was Wet-Chicken 2D, a two-dimensional version of the original Wet-Chicken benchmark [13]. In the original setting a canoeist

¹ Source code for the benchmark problem as well as a DUIPI implementation is available at <http://ahans.de/publications/icann2009/>

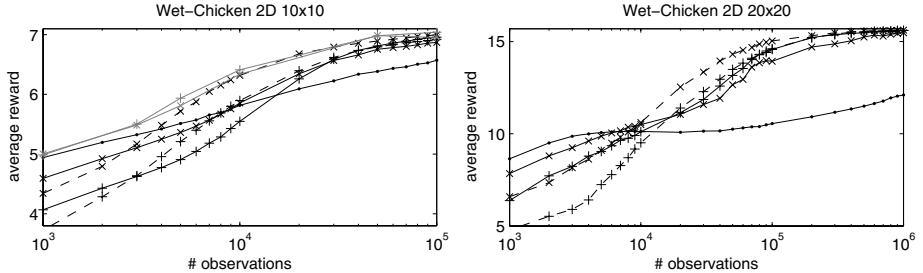


Fig. 1. Performance of policies generated using standard policy iteration ('●' marks), DUIPI (black lines), and the full-matrix method [10] (gray lines). $\xi = 0.5$ is indicated by 'x' marks, $\xi = 1$ by '+' marks. Solid lines represent policies generated using frequentist estimators, dashed lines represent policies generated using Bayesian estimation.

paddles on a one-dimensional river with length l and flow velocity $v = 1$. At position $x = l$ of the river there is a waterfall. Starting at position $x = 0$ the canoeist has to try to get as near as possible to the waterfall without falling down. If he falls down, he has to restart at position $x = 0$. The reward increases linearly with the proximity to the waterfall and is given by $r = x$. The canoeist has the possibility to drift ($x - 0 + v = x + 1$), to hold the position ($x - 1 + v = x$), or to paddle back ($x - 2 + v = x - 1$). River turbulence of size $s = 2.5$ causes the state transitions to be stochastic. Thus, after having applied the canoeist's action to his position (also considering the flow of the river), the new position is finally given by $x' = x + n$, where $n \in [-s, s]$ is a uniformly distributed random value. For the two-dimensional version the river is extended by a width w . Accordingly, there are two additional actions available to the canoeist, one to move the canoe to the left and one to move it to the right by one unit. The position of the canoeist is now denoted by (x, y) , the (re-)starting position is $(0, 0)$. The velocity of the flow v and the amount of turbulence s depend on y : $v = 3y/w$ and $s = 3.5 - v$. In the discrete problem setting, which we use here, x and y are always rounded to the next integer value. While on the left edge of the river the flow velocity is zero, the amount of turbulence is maximal; on the right edge there is no turbulence (in the discrete setting), but the velocity is too high to paddle back.

4.2 Results

We performed experiments with a river size of 10x10 (100 states) and 20x20 (400 states). For both settings a fixed number of observations was generated using random exploration. The observations were used as input to generate policies using the different algorithms. The discount factor was chosen as $\gamma = 0.95$. Each resulting policy was evaluated over 100 episodes with 1000 steps each. The results are summarized in fig. II (averaged over 100 trials). For clarity only the results of stochastic policies are shown (except for $\xi = 0$, i.e., standard policy iteration), they performed better than the deterministic ones in all experiments. Usually a method like DUIPI aims at quantile optimization, i.e., reducing the

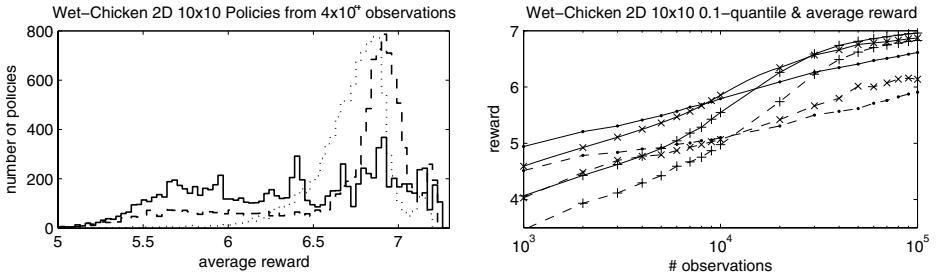


Fig. 2. Left: histograms of average rewards of 10^4 policies with $\xi = 0$ (solid), $\xi = 1$ (dashed), and $\xi = 2$ (dotted). For the generation of each policy 4×10^4 observations were used. Right: mean (solid) and 0.1-quantile (dashed) average rewards of policies with $\xi = 0$ ('•' marks), $\xi = 0.5$ ('x' marks), and $\xi = 1$ ('+' marks).

probability of generating very poor policies at the expense of a lower expected average reward. However, in some cases it is even possible to increase the expected performance, when the MDP exhibits states that are rarely visited but potentially result in a high reward. For Wet-Chicken states near the waterfall have those characteristics. An uncertainty unaware policy would try to reach those states if there are observations leading to the conclusion that the probability of falling down is low, which in fact is high. In [10] this is reported as “border-phenomenon”, which by our more general explanation is included. Due to this effect it is possible to increase the average performance using uncertainty aware methods for policy generation, which can be seen from the figure. For small numbers of observations and high ξ -values DUIPI performs worse as in those situations the action selection in the iteration is dominated by the uncertainty of the Q-values and not the Q-values themselves. This leads to a preference of actions with low uncertainty, the Q-values play only a minor role. This effect is increased by the fact that due to random exploration most observations are near the beginning of the river, where the immediate reward is low. Using a more intelligent exploration scheme could help to overcome this problem. Due to the large computational and memory requirements the full-matrix method could not be applied to the problem with river size 20x20. Moreover, results of the full-matrix version with Bayesian estimation are not shown as they would not have been distinguishable in the figure.

Fig. 2 compares uncertainty aware and unaware methods. Considering the uncertainty reduces the amount of poor policies and even increases the expected performance ($\xi = 0.5$). Setting $\xi = 1$ results in an even lower probability for poor policies at the expense of a lower expected average reward.

Table 1. Computation times to generate a policy using a single core of an Intel Core 2 Quad Q9550 processor

method	Wet-Chicken 5x5	Wet-Chicken 10x10	Wet-Chicken 20x20
full-matrix	5.61 s	1.1×10^3 s	—
DUIPI	0.002 s	0.034 s	1.61 s

5 Conclusion

In this paper, we presented DUIPI, a computationally very feasible algorithm for incorporation of uncertainty into the Bellman iteration. It only considers the diagonal of the covariance matrix encoding the covariance. While this causes the algorithm to be only approximate, it also decreases its complexity, decreasing the computational requirements by orders of magnitude. Moreover, we proposed a Bayesian parameter estimation that incorporates prior knowledge about the number of successor states. Our experiments show that DUIPI increases the robustness and performance of policies generated for MDPs whose exact parameters are unknown and estimated from only a fixed set of observations. In industrial applications observations are often expensive and arbitrary exploration not possible, we therefore believe that for those applications knowledge of uncertainty is crucial. Future work will consider application of UP to RL algorithms involving function approximation and utilizing knowledge of uncertainty for efficient exploration.

References

1. Calafiore, G., El Ghaoui, L.: On distributionally robust chance-constrained linear programs. In: Optimization Theory and Applications (2006)
2. D'Agostini, G.: Bayesian Reasoning in Data Analysis: A Critical Introduction. World Scientific Publishing, Singapore (2003)
3. Delage, E., Mannor, S.: Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In: Proc. of the Int. Conf. on Machine Learning (2007)
4. Engel, Y., Mannor, S., Meir, R.: Bayes meets Bellman: the Gaussian process approach to temporal difference learning. In: Proc. of the Int. Conf. on Machine Learning (2003)
5. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with Gaussian processes. In: Proc. of the Int. Conf. on Machine Learning (2005)
6. Ghavamzadeh, M., Engel, Y.: Bayesian policy gradient algorithms. In: Advances in Neural Information Processing Systems (2006)
7. Ghavamzadeh, M., Engel, Y.: Bayesian actor-critic algorithms. In: Proc. of the Int. Conf. on Machine Learning (2007)
8. Nilim, A., El Ghaoui, L.: Robustness in Markov decision problems with uncertain transition matrices. In: Advances in Neural Information Processing Systems (2003)
9. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons Canada, Ltd., Chichester (1994)
10. Schneegass, D., Udluft, S., Martinetz, T.: Uncertainty propagation for quality assurance in reinforcement learning. In: Proc. of the Int. Joint Conf. on Neural Networks (2008)
11. Strehl, A.L., Littman, M.L.: An empirical evaluation of interval estimation for markov decision processes. In: 16th IEEE Int. Conf. on Tools with Artificial Intelligence, pp. 128–135 (2004)
12. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
13. Tresp, V.: The wet game of chicken. Siemens AG, CT IC 4, Technical Report (1994)

Optimal Training Sequences for Locally Recurrent Neural Networks

Krzysztof Patan and Maciej Patan

Institute of Control and Computation Engineering,

University of Zielona Góra

{k.patan,m.patan}@issi.uz.zgora.pl

Abstract. The problem of determining an optimal training schedule for a locally recurrent neural network is discussed. Specifically, the proper choice of the most informative measurement data guaranteeing the reliable prediction of the neural network response is considered. Based on a scalar measure of the performance defined on the Fisher information matrix related to the network parameters, the problem was formulated in terms of optimal experimental design. Then, its solution can be readily achieved via the adaptation of effective numerical algorithms based on the convex optimization theory. Finally, some illustrative experiments are provided to verify the presented approach.

1 Introduction

A training of neural network, being the dynamic data-driven process requires a proper selection of measurement data to provide satisfactory representation of the modelled system behaviour [1][2]. In practice, this is equivalent to determination of a limited number of observational units obtained from the experimental environment in such a way as to obtain the best quality of the system responses.

The importance of input data selection has already been recognized in many application domains [3]. One of the most stimulating practical examples is Fault Detection and Identification (FDI) of industrial systems [4]. A crucial issue among the fundamental tasks of failure protection systems is to provide reliable diagnosis of the expected system state. To produce such a forecast, however, an accurate model is necessary and its calibration requires parameter estimation. Preparation of experimental conditions in order to gather informative measurements can be very expensive or even impossible (e.g. for the faulty system states). On the other hand, the data from real-world system may be very noisy and using all the available data may lead to significant systematic modelling errors. In result, we are faced with the problem of optimal choice of the available training data in order to obtain the most accurate model.

Although it is well known that the training quality for neural networks heavily depends on the choice of input sequences, surprisingly, there have been relatively few contributions to experimental design for those systems [5][6] and, in addition, they focus mainly on the multi-layer perceptron class of networks. The applicability of such a static type of networks for the modelling of dynamic systems

is rather limited. Recently, the problem of optimal selection of input sequences in the context of dynamic neural networks has been discussed by the authors in [74], where the problem is formulated in spirit of optimum experimental design theory for lumped systems [8]. However, the simulation results presented therein concern the training of the single dynamic neuron only. The contribution of this work is to extend this approach to the locally recurrent neural network with one hidden layer which can be applied in real-world systems. Moreover, to illustrate the delineated approach some experiments are performed using real process data.

2 Dynamic Neural Networks

The topology of the neural network considered is analogous to that of the multi-layered feedforward one and the dynamics are reproduced by the so-called dynamic neuron models [9,10,4]. The state-space representation of the neuron is shown in Fig. 1. In this paper a discrete-time dynamic network with n time varying inputs and m outputs is discussed. The description of such kind of a dynamic network with v hidden dynamic neurons, each containing an r -th order IIR filter, is given by the following nonlinear system:

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{Ax}(k) + \mathbf{Wu}(k) \\ \mathbf{y}(k) = \mathbf{C}\sigma(\mathbf{Bx}(k) + \mathbf{Du}(k) - \mathbf{g})^T \end{cases}, \quad (1)$$

where $N = v \times r$ represents the number of model states, $\mathbf{x} \in \mathbb{R}^N$ is the state vector, $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$ are input and output vectors, respectively, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the block diagonal state matrix ($\text{diag}(\mathbf{A}) = [\mathbf{A}_1, \dots, \mathbf{A}_v]$), $\mathbf{W} \in \mathbb{R}^{N \times n}$ ($\mathbf{W} = [\mathbf{w}_1 \mathbf{1}^T, \dots, \mathbf{w}_v \mathbf{1}^T]^T$, where \mathbf{w}_i is the input weight vector of the i -th hidden neuron), and $\mathbf{C} \in \mathbb{R}^{m \times v}$ are the input and output matrices, respectively, $\mathbf{B} \in \mathbb{R}^{v \times N}$ is a block diagonal matrix of feedforward filter parameters ($\text{diag}(\mathbf{B}) = [\mathbf{b}_1, \dots, \mathbf{b}_v]$), $\mathbf{D} \in \mathbb{R}^{v \times n}$ is the transfer matrix ($\mathbf{D} = [b_{01} \mathbf{w}_1^T, \dots, b_{0v} \mathbf{w}_v^T]^T$), $\mathbf{g} = [g_1 \dots g_v]^T$ denotes the vector of biases, and $\sigma : \mathbb{R}^v \rightarrow \mathbb{R}^v$ is the nonlinear vector-valued function. The presented structure can be viewed as a network with a single hidden layer containing v dynamic neurons as processing elements and an output layer with linear static elements. For structural details of the network considered, the interested reader is referred to [11].

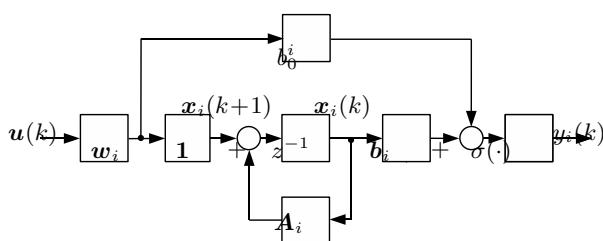


Fig. 1. State-space form of the i -th neuron with IIR filter

3 Optimal Sequence Selection Problem

3.1 Statistical Model

Let $\mathbf{y}^j = \mathbf{y}(\mathbf{u}^j; \boldsymbol{\theta}) = \{\mathbf{y}(k; \boldsymbol{\theta})\}_{k=0}^{L_j}$ denote the sequence of network responses for the sequence of inputs $\mathbf{u}^j = \{\mathbf{u}(k)\}_{k=0}^{L_j}$ related to the consecutive time instants $k = 0, \dots, L_j < \infty$ and selected from among an *a priori* given set of input sequences $\mathcal{U}_d = \{\mathbf{u}^1, \dots, \mathbf{u}^P\}$, where $\mathcal{U}_d \subset \mathcal{U}$. Here $\boldsymbol{\theta}$ represents a p -dimensional unknown network parameter vector which must be estimated using observations of the system (i.e. filter parameters, weights, slope and bias coefficients).

From the statistical point of view, the sequences of observations related to P input sequences may be considered as

$$\mathbf{z}^j(k) = \mathbf{y}^j(k; \boldsymbol{\theta}) + \boldsymbol{\varepsilon}^j(k), \quad k = 0, \dots, L_j, \quad j = 1, \dots, P, \quad (2)$$

where $\mathbf{z}^j(k)$ is the output and $\boldsymbol{\varepsilon}^j(k)$ denotes the measurement noise. It is customary to assume that the measurement noise is zero-mean, Gaussian and white, i.e.

$$E[\boldsymbol{\varepsilon}^i(k)\boldsymbol{\varepsilon}^j(k')] = v^2\delta_{ij}\delta_{kk'}, \quad (3)$$

where $v > 0$ is the standard deviation of the measurement noise, δ_{ij} and $\delta_{kk'}$ standing for the Kronecker delta functions.

An additional substantial assumption is that the training of the neural network, equivalent to the estimation of the unknown parameter vector $\boldsymbol{\theta}$, is performed via the minimization of the least-squares criterion

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \Theta_{\text{ad}}} \sum_{j=1}^P \sum_{k=0}^{L_j} \|\mathbf{z}^j(k) - \mathbf{y}^j(k; \boldsymbol{\theta})\|^2, \quad (4)$$

where Θ_{ad} is the set of admissible parameters. It becomes clear that since $\mathbf{y}^j(k; \boldsymbol{\theta})$ strongly depends on the input sequences \mathbf{u}^j it is possible to improve the training process through appropriate selection of input sequences.

3.2 Sequence Quality Measure and Experimental Design

In order to properly choose the input sequences which will be most informative for the training of the dynamic network, a quantitative measure of the goodness of parameter identification is required. A reasonable approach is to choose a performance measure defined on the Fisher Information Matrix (FIM), which is commonly used in optimum experimental design theory [12][8][13].

Sequences which guarantee the best accuracy of the least-squares estimates of $\boldsymbol{\theta}$ are then found by choosing \mathbf{u}^j , $j = 1, \dots, P$ so as to minimize some scalar measure of performance Ψ defined on the *average Fisher information matrix* given by [14]:

$$\mathbf{M} = \frac{1}{PL_j} \sum_{j=1}^P \sum_{k=0}^{L_j} \mathbf{H}(\mathbf{u}^j, k) \mathbf{H}^T(\mathbf{u}^j, k), \quad (5)$$

where

$$\mathbf{H}(\mathbf{u}, k) = \left(\frac{\partial y(\mathbf{u}, k; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right)_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \quad (6)$$

stands for the so-called *sensitivity matrix*, $\boldsymbol{\theta}^0$ being a prior estimate to the unknown parameter vector $\boldsymbol{\theta}$ which can be obtained from previous experiments or alternatively some known nominal values can be used [15][16][14][17].

Such a formulation is generally accepted in optimum experimental design for nonlinear dynamic systems, since the inverse of the FIM constitutes, up to a constant multiplier, the Cramér-Rao lower bound on the covariance matrix of any unbiased estimator of $\boldsymbol{\theta}$ [13], i.e.

$$\text{cov } \hat{\boldsymbol{\theta}} \succeq \mathbf{M}^{-1}. \quad (7)$$

Under somewhat mild assumptions [16][17], it is legitimate to assume that our estimator is *efficient* in the sense that the parameter covariance matrix achieves the lower bound.

As for criterion Ψ , various choices are proposed in the literature [13][8], but the most popular choice is so-called D-optimality (determinant) criterion:

$$\Psi(\mathbf{M}) = -\log \det \mathbf{M}; \quad (8)$$

which minimizes the volume of the uncertainty ellipsoid for the parameter estimates. The introduction of an optimality criterion renders it possible to formulate the sensor location problem as an optimization problem:

$$\Psi[\mathbf{M}(\mathbf{u}^1, \dots, \mathbf{u}^P)] \longrightarrow \min \quad (9)$$

with respect to \mathbf{u}^j , $j = 1, \dots, P$ belonging to the admissible set \mathcal{U} .

The direct consequence of the assumption (3) is that we admit replicated input sequences, i.e. some \mathbf{u}^j 's may appear several times in the optimal solution (because independent observations guarantee that every replication provides additional information). Consequently, it is sensible to reformulate the problem so as to operate only on the distinct sequences $\mathbf{u}^1, \dots, \mathbf{u}^S$ instead of $\mathbf{u}^1, \dots, \mathbf{u}^P$ by relabelling them suitably. To this end, we introduce r_1, \dots, r_S as the numbers of replicated measurements corresponding to the sequences $\mathbf{u}^1, \dots, \mathbf{u}^S$. In this formulation, the \mathbf{u}^i 's are said to be the *design* or *support* points, and p_1, \dots, p_S are called their weights. The collection of variables

$$\xi_P = \left\{ \begin{array}{l} \mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^S \\ p_1, p_2, \dots, p_S \end{array} \right\}, \quad (10)$$

where $p_i = r_i/P$, $P = \sum_{i=1}^S r_i$, is called the *exact design* of the experiment. The proportion p_i of observations performed for \mathbf{u}^i can be considered as the percentage of experimental effort spent at that sequence. Hence, we are able to rewrite the FIM in the form

$$\mathbf{M}(\xi_P) = \sum_{i=1}^S p_i \frac{1}{L_i} \sum_{k=0}^{L_i} \mathbf{H}^T(\mathbf{u}^i, k) \mathbf{H}(\mathbf{u}^i, k). \quad (11)$$

Here the p_i s are rational numbers, since both r_i s and P are integers. This leads to a discrete numerical analysis problem whose solution is difficult for standard optimization techniques, particularly when P is large. A potential remedy for this problem is to extend the definition of the design. This is achieved through the relaxation of constraints on weights, allowing the p_i s to be considered as real numbers in the interval $[0, 1]$. This assumption will be also made in what follows. Obviously, we must have $\sum_{i=1}^S p_i = 1$, so we may think of the designs as probability distributions on \mathcal{U} . This leads to the so-called *continuous* designs, which constitute the basis of the modern theory of optimal experiments [8,13]. It turns out that such an approach drastically simplifies the design, and the existing rounding techniques [8] justify such an extension. Thus, we shall operate on designs of the form

$$\xi = \left\{ \mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^S; \quad p_1, p_2, \dots, p_S; \quad \sum_{i=1}^S p_i = 1; \quad \forall i \quad p_i \geq 0 \right\}, \quad (12)$$

which concentrates Pp_1 observational sequences for \mathbf{u}^1 (so we repeat approximately Pp_1 times the presentation of this sequence during the training of the network), Pp_2 for \mathbf{u}_2 , and so on. Then we may redefine optimal design as a solution to the optimization problem

$$\xi^* = \arg \min_{\xi \in \Xi(\mathcal{U})} \Psi[\mathbf{M}(\xi)], \quad (13)$$

where $\Xi(\mathcal{U})$ denotes the set of all probability distributions on \mathcal{U} .

3.3 Characterization of Optimal Solutions

In the remainder of this chapter we shall assume that $\mathbf{H} \in C(\mathcal{U}; \mathbb{R}^p)$. The following characterizations of the optimal design ξ^* can be derived in a rather straightforward manner from the general results given in [14] or [17].

Theorem 1. *An optimal design exists comprising no more than $p(p+1)/2$ support sequences. Moreover, the set of optimal designs is convex.*

The practical importance of this property cannot be underestimated since we can restrict our attention to the designs with limited number of sequences what significantly reduces the complexity of resulting optimization problem. But the next theorem is essential for the approach considered and provides a tool for checking the optimality of designs. It is usually called an *equivalence theorem* [18].

Theorem 2 (Equivalence theorem). *The following conditions are equivalent:*

- (i) *the design ξ^* minimizes $\Psi(\mathbf{M}) = -\ln \det \mathbf{M}(\xi)$,*
- (ii) *the design ξ^* minimizes $\max_{\mathbf{u}^i \in \mathcal{U}} \phi(\mathbf{u}^i, \xi)$, and*
- (iii) *$\max_{\mathbf{u}^i \in \mathcal{U}} \phi(\mathbf{u}^i, \xi) = p$,*

and the so-called *sensitivity function*

$$\phi(\mathbf{u}^i, \xi) = \text{trace} \left(\frac{1}{L_i} \sum_{k=0}^{L_i} \mathbf{H}^T(\mathbf{u}^i, k) \mathbf{M}^{-1} \mathbf{H}(\mathbf{u}^i, k) \right)$$

is of paramount importance here as it can be interpreted in terms of average variance of the estimated system response being the natural measure for the quality of the training process. From the result above it comes immediately that suppressing the maximal level of the prediction variance is equivalent to the optimization of the D-optimality criterion. This paves the way to almost direct application of numerous efficient algorithms known from experimental design theory to the discussed problem. Since analytical determination of optimal designs is difficult or impossible even for very simple network structures, some iterative design procedures will be required. A simple computational scheme for that purpose is given in the next section.

4 Selection of Training Sequences

In the case considered in the paper, a computational algorithm can be derived based on the mapping $\mathcal{T} : \Xi(\mathcal{U}) \rightarrow \Xi(\mathcal{U})$ defined by

$$\mathcal{T}\xi = \left\{ \begin{array}{c} \mathbf{u}^1, \dots, \mathbf{u}^S \\ p_1\phi(\mathbf{u}^1, \xi)/p, \dots, p_S\phi(\mathbf{u}^S, \xi)/p \end{array} \right\}. \quad (14)$$

From Theorem 2 it follows that a design ξ^* is D-optimal if it is a fixed point of the mapping \mathcal{T} , i.e.

$$\mathcal{T}\xi^* = \xi^*. \quad (15)$$

Therefore, the following algorithm can be used as a generalization of that proposed in [19, p. 139] for the classical optimum experimental design problem consisting in iterative computation of a D-optimum design on a finite set:

Step 1. Guess a discrete starting design $\xi^{(0)}$ such that $p_i^{(0)} > 0$ for $i = 1, \dots, S$.

Choose some positive tolerance $\eta \ll 1$. Set $\ell = 0$.

Step 2. If the condition

$$\frac{\phi(\mathbf{u}^i, \xi^{(\ell)})}{p} < 1 + \eta, \quad i = 1, \dots, S$$

is satisfied, then *STOP*.

Step 3. Construct the next design $\xi^{(k+1)}$ by determining its weights according to the rule

$$p_i^{(\ell+1)} = p_i^{(\ell)} \frac{\phi(\mathbf{u}^i, \xi^{(\ell)})}{m}, \quad i = 1, \dots, S,$$

increment k by one and go to Step 2.

The convergence result of this scheme can be found in [17].

5 Illustrative Example

Simulation setting. All experiments were carried out using the AMIRA DR300 laboratory system. This laboratory system is used to control the rotational speed of a DC motor with a changing load [4]. A separately excited DC motor was modelled by using the dynamic neural network presented briefly in Section 2. The output signal was the rotational speed (T) measured by an analog tachometer. The input signal (U) was selected as a sum of sinusoids:

$$U(k) = 3 \sin(2\pi 1.7k) + 3 \sin(2\pi 1.1k - \pi/7) + 3 \sin(2\pi 0.3k + \pi/3) \quad (16)$$

The structure of the neural network model (II) was selected arbitrarily and had the following structure: one input, three IIR neurons with second order filters and hyperbolic tangent activation functions, and one linear output neuron. Taking into account that a neural network is a redundant system, some of its parameters are not identifiable. In order to apply optimum experimental design to the neuron training, certain assumptions should be made. So, without loss of generality, let us assume that the feedforward filter parameter b_0 for each hidden neuron is fixed to the value of 1. This reduces the dimensionality of estimation and assures the identifiability of the rest of the parameters (i.e. it assures that the related FIM is non-singular).

At the beginning, the network was preliminarily trained in order to obtain the initial parameters estimates. Feeding the laboratory system with signal (II), a learning set containing 500 samples was formed, and then the training process was carried out off-line for 2000 steps using the Extended Dynamic Back-Propagation (EDBP) algorithm [7]. At the second stage of the training, the learning data were split into 20 time sequences, containing 150 consecutive samples each. The design purpose was to choose from this set of all learning patterns the most informative sequences (in the sense of D-optimality) and their presentation frequency (i.e. how often they should be repeated during the training). To determine the optimal design, a numerical routine from Section 4 was implemented in the form of the MATLAB program. All the admissible learning sequences taken with equal weights formed the initial design. The accuracy of the design algorithm was set to $\eta = 10^{-2}$.

Results. The network was preliminarily trained and the initial network parameters estimates are presented in the second column of Table II. In this case the Sum of Squared Errors (SSE) calculated using the training set was equal to 5.7001. After that, the training of the network was continued in two ways. The first way was to use the optimal training sets selected during the optimum experimental design phase. The second way was to use random sequences as the training ones. The purpose of these experiments is to check the quality of parameter estimation. In the case considered here the optimal design consists of the sequences 5, 6, 8 and 16. For a selected design, each distinct sequence was replicated proportionally to its weight in the design with total number of replications assumed to be $P = 10$. For example, if the optimal design consists of the four aforementioned

sequences with the weights 0.3, 0.1, 0.3 and 0.3, respectively, then during the training the 5-th, 8-th and 16-th sequences were used three times each, and the 6-th sequence only once (in random order). The training procedure was repeated 10 times using different measurement noise affecting the output of the system. The statistics are presented in Table 1. As we can see there, the application of training sets selected according to the optimal design leads to the better accuracies of parameter estimates than in the case of randomly selected training sets. It is observed that the standard deviation of each network parameter has lower value in the case of the optimal design what means the more reliable estimate of a given parameter.

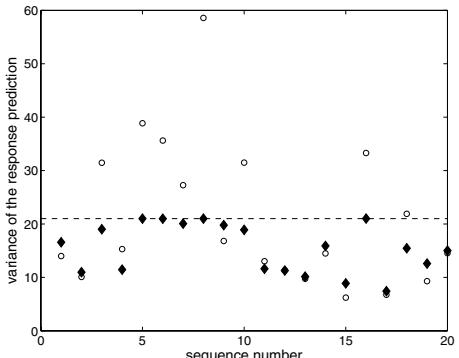
The uncertainty of the network response prediction is examined based on the parameter estimates determined using the optimal and random designs. The testing phase of each of 10 realizations of locally recurrent network was performed using 1000 samples different from the training ones, and for each realization the quality measure in the form of the SSE was calculated. The results of testing are presented in Table 2. Looking at these results one can state that using random design it is possible to obtain a good generalization of the network, e.g. networks 9 and 10, but the results of training are not repetitive as in the case of optimal design when 9 of 10 training run give the similar good results. This fact, in connection with the plot of response prediction variance (Fig. 2) clearly shows

Table 1. Sample mean and the standard deviation of parameter estimates

para- meter	initial value	sample mean		standard deviation	
		random design	optimal design	random design	optimal design
w_1	0.3232	0.2867	0.2894	0.0104	0.0028
w_2	0.9000	0.9105	0.9082	0.0034	0.0009
w_3	0.0758	0.0898	0.0789	0.0194	0.0027
b_{11}	0.8328	0.8187	0.8195	0.0040	0.0011
b_{21}	-0.6316	-0.6053	-0.6072	0.0078	0.0019
b_{31}	0.8558	0.8616	0.8581	0.0079	0.0011
b_{12}	0.7892	0.7742	0.7747	0.0042	0.0011
b_{22}	0.0631	0.0910	0.0897	0.0082	0.0019
b_{32}	0.5745	0.5808	0.5812	0.0076	0.0011
a_{11}	0.1258	0.1302	0.1301	0.0012	0.0003
a_{21}	0.0853	0.0807	0.0812	0.0015	0.0004
a_{31}	-0.4171	-0.4196	-0.4170	0.0055	0.0015
a_{12}	0.1656	0.1703	0.1703	0.0012	0.0003
a_{22}	0.0266	0.0217	0.0221	0.0016	0.0004
a_{32}	-0.5566	-0.5587	-0.5562	0.0052	0.0015
g_1	-0.3794	-0.4057	-0.4024	0.0132	0.0055
g_2	-0.3978	-0.3599	-0.3673	0.0206	0.0089
g_3	0.3187	0.3040	0.3136	0.0189	0.0008
c_1	-0.4908	-0.4905	-0.4893	0.0081	0.0032
c_2	0.7773	0.7708	0.7716	0.0078	0.0035
c_3	0.4540	0.4438	0.4408	0.0075	0.0006

Table 2. Results of network testing – SSE measure

Network realization	Random design	Optimal design
1	29.7367	31.0998
2	27.4287	26.5564
3	42.4463	26.4758
4	85.8052	26.6182
5	99.5833	26.4214
6	82.9475	26.4577
7	35.3615	26.6521
8	29.6130	26.5550
9	26.8438	26.5030
10	26.2403	26.2885

**Fig. 2.** Variances of the model response prediction for the optimum design (diamonds) and random design (circles)

that training based on optimal learning sequences leads to greater reliability of the network response as the maximal variance level can be significantly reduced.

Taking into account the computation burden, for the case considered here the selection of the optimal plan lasted 9 seconds. For comparison, the 500 training steps carried out off-line with the sequence of the length 500 lasted 117.18 seconds. Summarizing, the process of the optimal design selection by itself does not significantly prolongate the overall training procedure.

6 Conclusions

The results reported in this paper show that some well-known methods of optimum experimental design for linear regression models can be easily extended to the setting of the optimal training sequence selection problem for dynamic neural networks. The clear advantage of the proposed approach is that the quality of the training process measured in terms of the uncertainty of network response prediction can be significantly improved with the same effort spent on training or, alternatively, training process complexity can be reduced without degrading the network performance.

The proposed approach was also tested using other network structures. Experiments were carried out for a locally recurrent network with two hidden neurons as well as for a network with five hidden neurons. In each case considered, the results are similar to these presented in the paper taking into account the reliability of parameters estimates.

Future research will be focused on the application of other methods for determining optimal designs, namely methods which do not determine the presentation frequency, thus its practical application is easier.

References

1. Gupta, M.M., Jin, L., Homma, N.: Static and Dynamic Neural Networks. From Fundamentals to Advanced Theory. John Wiley & Sons, New Jersey (2003)
2. Korbicz, J., Kościelny, J., Kowalcuk, Z., Cholewa, W.: Fault Diagnosis. Models, Artificial Intelligence, Applications. Springer, Heidelberg (2004)
3. van de Wal, M., de Jager, B.: A review of methods for input/output selection. *Automatica* 37, 487–510 (2001)
4. Patan, K.: Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes. LNCIS. Springer, Berlin (2008)
5. Fukumizu, K.: Statistical active learning in multilayer perceptrons. *IEEE Transactions on Neural Networks* 11, 17–26 (2000)
6. Witczak, M.: Toward the training of feed-forward neural networks with the D-optimum input sequence. *IEEE Transactions on Neural Networks* 17, 357–373 (2006)
7. Patan, K., Patan, M.: Selection of training sequences for locally recurrent neural network training. In: Malinowski, K., Rutkowski, L. (eds.) Recent Advances in Control and Automation, pp. 252–262. Academic Publishing House, EXIT, Warsaw (2008)
8. Fedorov, V.V., Hackl, P.: Model-Oriented Design of Experiments. Lecture Notes in Statistics. Springer, New York (1997)
9. Tsoi, A.C., Back, A.D.: Locally recurrent globally feedforward networks: A critical review of architectures. *IEEE Transactions on Neural Networks* 5, 229–239 (1994)
10. Marcu, T., Mirea, L., Frank, P.M.: Development of dynamical neural networks with application to observer based fault detection and isolation. *International Journal of Applied Mathematics and Computer Science* 9(3), 547–570 (1999)
11. Patan, K.: Stability analysis and the stabilization of a class of discrete-time dynamic neural networks. *IEEE Transactions on Neural Networks* 18, 660–673 (2007)
12. Atkinson, A.C., Donev, A.N.: Optimum Experimental Designs. Clarendon Press, Oxford (1992)
13. Walter, E., Pronzato, L.: Identification of Parametric Models from Experimental Data. Springer, London (1997)
14. Patan, M.: Optimal Observation Strategies for Parameter Estimation of Distributed Systems. Lecture Notes in Control and Computer Science, vol. 5. Zielona Góra University Press, Zielona Góra (2004)
15. Uciński, D.: Optimal selection of measurement locations for parameter estimation in distributed processes. *International Journal of Applied Mathematics and Computer Science* 10(2), 357–379 (2000)
16. Rafajłowicz, E.: Optimum choice of moving sensor trajectories for distributed parameter system identification. *International Journal of Control* 43(5), 1441–1451 (1986)
17. Uciński, D.: Optimal Measurement Methods for Distributed Parameter System Identification. CRC Press, Boca Raton (2005)
18. Kiefer, J., Wolfowitz, J.: Optimum designs in regression problems. *The Annals of Mathematical Statistics* 30, 271–294 (1959)
19. Pázman, A.: Foundations of Optimum Experimental Design. Mathematics and Its Applications. D. Reidel Publishing Company, Dordrecht (1986)

Statistical Instance-Based Ensemble Pruning for Multi-class Problems

Gonzalo Martínez-Muñoz^{1,2}, Daniel Hernández-Lobato¹, and Alberto Suárez¹

¹ Universidad Autónoma de Madrid, EPS, Madrid 28049 Spain

² Oregon State University, Corvallis, OR 97331 USA

Abstract. Recent research has shown that the provisional count of votes of an ensemble of classifiers can be used to estimate the probability that the final ensemble prediction coincides with the current majority class. For a given instance, querying can be stopped when this probability is above a specified threshold. This instance-based ensemble pruning procedure can be efficiently implemented if these probabilities are pre-computed and stored in a lookup table. However, the size of the table and the cost of computing the probabilities grow very rapidly with the number of classes of the problem. In this article we introduce a number of computational optimizations that can be used to make the construction of the lookup table feasible. As a result, the application of instance-based ensemble pruning is extended to multi-class problems. Experiments in several UCI multi-class problems show that instance-based pruning speeds-up classification by a factor between 2 and 10 without any significant variation in the prediction accuracy of the ensemble.

Keywords: Instance based pruning, ensemble learning, neural networks.

1 Introduction

Ensemble methods generate a collection of diverse classifiers by introducing variations in the algorithm used to train the base predictors or in the conditions under which learning takes place [12,34,5]. The classification of an unlabeled instance by the ensemble is obtained by combining the predictions of the individual classifiers. In majority voting, each classifier in the ensemble is asked to predict the class label of the instance considered. Once all the classifiers have been queried, the class that receives the greatest number of votes is returned as the final decision of the ensemble. The time needed to classify an instance increases linearly with the size of the ensemble. However, in many cases, it is not necessary to compute the predictions of every classifier to obtain a reliable estimate of the prediction of the complete ensemble. Assuming that the classifiers of the ensemble are generated independently when conditioned to the training data¹, the class labels predicted by the ensemble members can be seen as independent identically distributed random variables. With this assumption, it is possible to compute the probability that the current majority class coincides with

¹ Note that this is different from assuming that the classifiers are unconditionally independent.

the final ensemble prediction, on the basis of the known class votes. Therefore, for a given instance, the querying process can be halted when this probability is above a specified threshold. This dynamical instance-based ensemble pruning can be used in practice to speed-up the classification process [67].

For statistical IB-pruning to be effective, the probability that the final decision of the ensemble will not change when taking into account the remaining votes needs to be rapidly computed. A possible implementation is to store pre-computed values of these probabilities in a lookup table. The difficulty is that both the size of the table and the cost of evaluating the probability values grow very quickly with the number of class labels. Hence, applying IB-pruning to classification problems with more than two classes remains a challenging task.

In this work we introduce several techniques that can be applied to reduce the size of the lookup table and to optimize the process of computing these probability values. With the proposed optimizations, IB-pruning can be effectively used to reduce the time of classification by ensembles also in multi-class problems. Specifically, in this work problems with up to 11 different classes are considered. In the problems investigated, the number of classifiers queried is reduced by a factor between ≈ 2 and ≈ 10 . The empirical rates of disagreement between the class predicted by the dynamically pruned ensembles and the complete ones are close to the confidence level specified and often below it. Furthermore, the actual differences in classification error between these two ensembles are very small; in fact, much smaller than the disagreement rates.

The article is structured as follows: Statistical instance-based ensemble pruning is briefly reviewed in Section 2. Section 3 describes the optimizations that make the application of IB-pruning to problems with more than two classes practicable. Section 4 illustrates the improvements in classification speed that can be achieved when IB-pruning is used in a variety of multi-class problems. Finally, the conclusions of this work are summarized in Section 5.

2 Statistical Instance-Based Ensemble Pruning

Consider an ensemble composed of T classifiers $\{h(\mathbf{x})_i\}_{i=1}^T$. The class assigned to an unlabeled instance, \mathbf{x} , when majority voting is used to combine the outputs of the ensemble classifiers is

$$\arg \max_y \sum_{t=1}^T \mathcal{I}(h_t(\mathbf{x}) = y), \quad y \in \mathcal{Y}, \quad (1)$$

where $h_t(\mathbf{x})$ is the prediction of the t -th ensemble member, \mathcal{I} is an indicator function and $\mathcal{Y} = \{y_1, \dots, y_l\}$ is the set of possible class labels.

The predictions of the classifiers in the ensemble can be summarized in the vector $\mathbf{T} \equiv \{T_1, T_2, \dots, T_l; \sum_{i=1}^l T_i = T\}$ where T_i is the number of members in the ensemble that predict class y_i for the instance to be classified. Similarly, the vector that stores the prediction of the first $t \leq T$ classifiers in the ensemble is $\mathbf{t} \equiv \{t_1, t_2, \dots, t_l; \sum_{i=1}^l t_i = t; t_i \leq T_i, i = 1, 2, \dots, l\}$. Assuming the individual classifiers of the ensemble are built independently when conditioned to the

training data, the probability that the class labels predicted by the subensemble of size $t < T$ and by the complete ensemble of size T coincide is

$$\tilde{\mathcal{P}}(\mathbf{t}, T) = \sum_{\mathbf{T}}^* \left[\frac{(T-t)!}{(t+l)_{T-t}} \prod_{i=1}^l \frac{(t_i+1)_{T_i-t_i}}{(T_i-t_i)!} \right], \quad (2)$$

where $(a)_n = a(a+1)\cdots(a+n-1)$ is the Pochhammer symbol, or rising factorial, with a and n nonnegative integers, and the asterisk indicates that the summation runs over all values of \mathbf{T} such that $\sum_{i=1}^l T_i = T$, $\{T_i \geq t_i, i = 1, 2, \dots, l\}$, and $T_{k_t^*} > T_j$ for $j \neq k_t^*$, where k_t^* is the majority class after querying the first t classifiers. See [6] for further details.

If it is acceptable that the coincidence between these two predictions is not necessarily certain, but occurs with a high confidence level π , (2) can be used to stop the querying process after the predictions of t classifiers in the ensemble are known, when the vector of class predictions of the current subensemble \mathbf{t} is such that $\tilde{\mathcal{P}}(\mathbf{t}, T) \geq \pi$. Since the fraction of examples that are assigned a different class label by the pruned ensemble of size t and the complete subensemble is at most $1 - \pi$, the differences in error should be smaller than this disagreement rate. In practice, the changes in class labels affect both correctly labeled examples and misclassified examples in approximately equal numbers, and the differences in error rates are much smaller than this upper bound. Therefore, for a given instance the partial ensemble prediction can be used instead of the complete ensemble to save time in the classification process at the expense of small differences in the assignment of class labels, which generally do not translate into large differences in generalization performance.

To determine whether the polling process can be stopped it is necessary to know the values of $\tilde{\mathcal{P}}(\mathbf{t}, T)$, whose computation can be costly. Since for a given number of classes l and ensemble size T , these probabilities only depend on \mathbf{t} , they can be pre-computed and stored in a lookup table. The difficulty is that for $l < T$ both the size of the lookup table that stores pre-computed probability values and the cost of computing each entry in the table are nearly exponential in the number of different class labels l . The number of different entries in this table is $\binom{T+l}{T}$, namely the number of different ways in which T objects can be assigned to $l+1$ classes. The extra class corresponds to the unknown predictions of the classifiers that have not been queried.

The cost of evaluating (2) depends on the number of vectors \mathbf{T} involved in the summation. This number is different for each entry of the table and is given by the number of values of \mathbf{T} such that $T_{k_t^*}$ is the majority class. As a consequence of the constraints $T_i \geq t_i$ its value decreases as the value of the components of \mathbf{t} increase. Thus, the maximum number of different vectors \mathbf{T} , $C_{max}(T, l)$, is obtained for the entry corresponding to $\mathbf{t} = \mathbf{0}$. This is the worst possible scenario which sets an upper bound on the time-complexity of evaluating (2).

The value of $C_{max}(T, l)$ can be expressed in terms of $N(n, r, m)$, the number of different ways of decomposing a positive integer n into r non-negative integers that are all smaller or equal than m ,

$$n_1 + n_2 + \cdots + n_r = n \quad n_i \leq m \quad i = 1, 2, \dots, r.$$

If $m \geq n$ the problem reduces to computing the different ways of assigning n objects to r classes with repetitions allowed, namely $\binom{n+r-1}{n}$. Otherwise, it is given by the formula

$$N(n, r, m) = \begin{cases} \sum_{i=0}^{\lfloor n/(m+1) \rfloor} (-1)^i \binom{r}{i} \binom{n - i(m+1) + r - 1}{n - i(m+1)} & \text{if } n \leq mr \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

To calculate $C_{max}(T, l)$ we need to count the number of different vectors \mathbf{T} involved in the summation in which the minority votes can be distributed among the $l-1$ minority classes. Assume that the majority class gets i votes. The number of different ways of distributing the remaining votes among the remaining $l-1$ classes, so that all of the minority classes receive less than i votes is $N(T-i, l-1, i-1)$. Summing over all possible values of i , we obtain

$$C_{max}(T, l) = \sum_{i=\lceil T/l \rceil}^T N(T-i, l-1, i-1). \quad (4)$$

Figure 1 (left) displays in log scale the size of the table and the maximum number of operations needed to calculate each element in the table ($C_{max}(T, l)$) as the number of class labels l increases for $T = 101$. These two numbers grow very rapidly with the number of classes l . The increase from $l-1$ classes to l classes corresponds in both cases to a multiplicative factor of $\approx T/l$. In consequence, the problem quickly becomes intractable even for relatively small numbers of

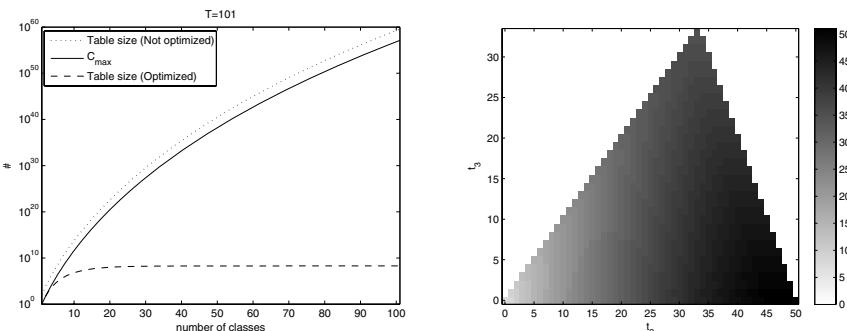


Fig. 1. (Left) Size of the lookup table (with and without optimizations) and complexity of the computation of the table entries ($C_{max}(T, l)$) as a function of number of classes of the lookup or $T = 101$. (Right) Lookup table for $T = 101$, $\pi = 0.99$ and $l = 3$.

classes. For $T = 101$ and $l = 2, \dots, 7$, the maximum number of vectors \mathbf{T} involved in the summation (and table sizes) are: 51 (5253), 1734 (182104), 44867 (4780230), 937747(101340876),16489227(1807245622), 250881300(27883218168), respectively. Thus, even for $l = 3$, building the lookup table is costly: The table has 182,104 entries and computing each entry demands as much as 1,734 operations whose time-complexity is $\mathcal{O}(l)$.

3 Optimizations

In this section we describe several exact optimizations that can be applied to build the lookup table in a more efficient way¹. First, each of the terms in the summation in (2) can be computed faster if the factors that are repeatedly used in the calculation are stored in auxiliary tables. Second, the size of the lookup table can also be reduced by storing the minimal amount of information that is needed to reach a decision on when to stop querying classifiers. Finally, some of the terms contribute with the same value to the sum in (2). Therefore, it is sufficient to compute this value only once and then multiply the result by the number of equivalent terms.

The calculation of each term inside the summation in (1) requires the computation of the product of one factor of the form

$$(T - t)! / (t + l)_{T-t} . \quad (5)$$

and of l factors of the form

$$(t_i + 1)_{T_i - t_i} / (T_i - t_i)! \quad (6)$$

The number of different values that these factors can have is determined by T_i , t_i and t . The variables T_i and t take values in the range $[0, T]$ and $t_i \leq T_i$. Therefore, for a given T there are only $(T + 1)T/2$ different outcomes for (6) and $T + 1$ for (5). This is a fairly small number of values that can be easily pre-computed and stored in an auxiliary table. Thus, each term in the summation that appears in (2) can be readily computed as a product of l of these factors. In addition, to avoid numerical difficulties with the computations that involve factorials, (6) and (5) are calculated using the prime-factor decomposition of the numbers involved.

Regarding the reduction of the size of the lookup table, the following optimizations can be made: Instead of storing probabilities, it is sufficient to generate a table whose entries correspond to the distribution of votes in the minority classes only. The value stored in the table for each entry is the minimum number of votes for the majority class needed to estimate the complete ensemble prediction with a confidence level π , given the observed minority classes. The size of the lookup table can be further reduced if the votes in \mathbf{t} are kept in memory sorted in decreasing order. The overhead associated to keeping the votes sorted when new

¹ Source code available at <http://www.eps.uam.es/~gonzalo/publications/>

predictions are known is very small, of $O(l)$ per classifier queried. Thus, the first element in \mathbf{t} corresponds to the currently observed majority class. The remaining observations correspond to the minority classes which are used as indexes for the lookup table. Because $t_i \geq t_{i+1}$ for all i , the value of t_i is at most $\lceil T/i \rceil - 1$ for $i = 1, 2, \dots, l$.

The dependence of the size of the table on the number of classes with and without the optimizations described is shown in the left plot of Fig. 11 for an ensemble of size $T = 101$. The right plot of Fig. 11 displays the lookup table for $l = 3$, $T = 101$ and $\pi = 99\%$. The value stored in the position of the table labeled by the pair (t_2, t_3) corresponds to the minimum number of votes of the majority class $t_1^*(t_2, t_3)$ that are needed to anticipate the ensemble prediction with a confidence level $\pi = 99\%$ when the two minority classes have received t_2 and t_3 votes, respectively. The triangular shape of the table is given by the constraints $t_1 > t_2 \geq t_3$. The critical values of the majority class are plotted using an inverted gray scale: a darker color indicates that higher values for the majority class are needed to stop the voting process, according to the scale displayed on the right-hand side of the figure.

The optimizations introduced thus far allow to compute lookup tables for small and intermediate values of l , the total number of classes. In problems with a larger number classes, the rapid growth in the number of terms that need to be considered makes the calculation of (2) unfeasible. Nevertheless, this computation is manageable for instances that during the classification process have received votes in at most k classes. Instances that have votes in more than k classes can be classified by querying all the classifiers in the ensemble, without pruning. An instance that has received votes in only k classes when t classifiers have been queried is characterized by a vector \mathbf{t} whose $l - k$ last components are zero: $t_{k+1} = t_{k+2} = \dots = t_l = 0$. For these instances, it is sufficient to compute a lookup table with the same dimensions as a table designed for a problem with k classes. Note that if $t_i = 0$ then (6) is equal to one independently of the value of T_i . This observation can be used to simplify the computation of (2) in two ways: First, in each of the terms in the summation (2) it is not necessary to multiply by the factors of the form (6) for classes $i = k + 1, k + 2, \dots, l$, because they are all equal to one. Second, when $\mathbf{t} = \{t_1, t_2, \dots, t_k, 0, \dots, 0\}$ all terms \mathbf{T} of the form $\{T_1, \dots, T_k, *, \dots, *\}$ contribute with the same value in (2). For a particular value of T_1, T_2, \dots, T_k , the number of terms \mathbf{T} of the form $\{T_1, \dots, T_k, *, \dots, *\}$, is $N(T - \sum_{i=1}^k T_i, l - k, T_1 - 1)$; that is, the number of ways in which the remaining votes, $T - \sum_{i=1}^k T_i$, can be decomposed among the remaining $l - k$ classes so that none of these classes receives more votes than T_1 . Thus, the terms of the form $\{T_1, \dots, T_k, *, \dots, *\}$ can be grouped and their contribution in (2) calculated by computing the value of one of them and then multiplying by their count $N(T - \sum_{i=1}^k T_i, l - k, T_1 - 1)$. Using this optimization the actual number of different terms that need to be computed in (2) grows slowly with l for fixed k . For example, for $T = 101$ and $k = 3$ the number of final vectors \mathbf{T} to be computed grow from 44867 with $l = 4$ to 59821 with $l = 101$.

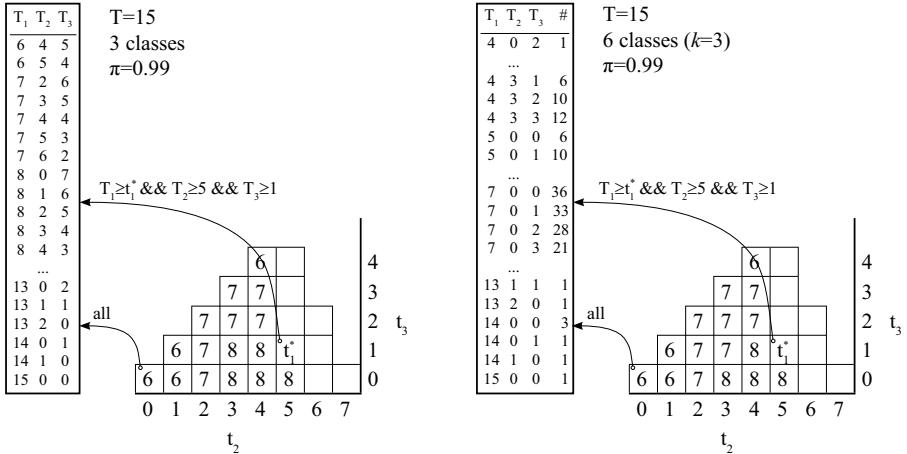


Fig. 2. Lookup table computation process for $T = 15$ and $l = 3$ (left) and for $T = 15$, $l = 6$ and $k = 3$ (right)

Fig. 2 illustrates the process of construction of the lookup table for $T = 15$, $l = 3$ and $k = 3$ (left part of the figure) and for $T = 15$, $l = 6$ and $k = 3$ (right part of the figure). On the left-hand side the list of final vectors \mathbf{T} that have to be analyzed during the computation of the lookup table are shown. Note that not all of them have to be taken into account to compute every entry of the table. For instance, to calculate the entry $(t_2 = 0, t_3 = 0)$ all combinations (T_1, T_2, T_3) are needed. For entry $(t_2 = 5, t_3 = 1)$ only combinations such that $T_2 \geq 5$ and $T_3 \geq 1$ are used. In the case of $l = 6$ classes and $k = 3$ (right part of the figure) the procedure for computing the table is equivalent except that each term is multiplied by the number of equivalent vectors \mathbf{T} that contribute with the same value (fourth column of the list of combinations, under the header #).

4 Experiments

The performance of the instance-based pruning procedure described is illustrated in experiments on benchmark multi-class problems from [8]. For each problem, we generate 100 random partitions of the data into two disjoint sets that contain 2/3 and 1/3 of the instances, respectively. The first set is used for constructing the ensembles while the second one is used for evaluation. For each of these partitions, we build a bagging ensemble composed of 101 neural networks [1]. In bagging, each network is trained on a different bootstrap sample of the data available. The neural networks are single layer feed forward networks with 5 units in the hidden layer and *soft-max* outputs. They are trained using a maximum of 1,000 epochs using the quasi-Newton method BFGS. These choices were made so that a good overall accuracy is obtained in the classification problems investigated. The *nnet* R package [9] is used to train the neural networks. The performance of the ensembles is estimated on the test set. For each instance

Table 1. Results for the analyzed datasets

Dataset	classes (k)	Disagreement (in %)	Classification Error IB-Pruning	Classification Error No pruning	# of NN	(Q1 Q2 Q3)
Dermatology	6	0.034±0.166	2.82±1.31	2.80±1.31	10.9	(8.0 8.0 10.0)
DNA	3	0.090±0.108	5.20±0.68	5.19±0.68	12.0	(7.0 7.0 9.0)
E-coli	8	0.196±0.485	23.84±11.22	23.80±11.27	34.7	(8.0 11.0 63.0)
	$(k = 5)$ 18.48% instances with votes for more than 5 classes					
Glass	6	0.310±0.651	31.18±5.01	31.17±4.93	31.4	(10.0 18.0 45.0)
Iris	3	0.080±0.394	4.78±2.59	4.82±2.68	9.8	(7.0 7.0 7.0)
LED24	10	0.149±0.139	28.40±1.79	28.40±1.79	21.9	(8.0 8.0 14.0)
	$(k = 5)$ 9.24% instances with votes for more than 5 classes					
New-Thyroid	3	0.056±0.274	3.94±2.13	3.94±2.11	9.7	(7.0 7.0 7.0)
Satellite	6	0.128±0.068	12.23±0.68	12.23±0.68	14.8	(8.0 8.0 10.0)
Segment	7	0.042±0.094	2.91±0.55	2.90±0.55	11.6	(8.0 8.0 8.0)
	$(k = 5)$ 0.74% instances with votes for more than 5 classes					
Vehicle	4	0.408±0.395	17.82±2.04	17.79±2.01	24.1	(7.0 11.0 29.0)
Vowel	11	0.364±0.305	23.32±2.44	23.24±2.40	50.2	(14.0 31.0 101.0)
	$(k = 5)$ 29.99% instances with votes for more than 5 classes					
Waveform	3	0.284±0.169	16.87±1.28	16.85±1.28	18.7	(7.0 9.0 17.0)
Wine	3	0.068±0.334	2.19±1.89	2.15±1.91	9.4	(7.0 7.0 7.0)

to be classified, the networks in the ensemble are queried at random without repetition. The vector of votes \mathbf{t} is updated so that its elements remain sorted. The querying process is stopped when t_1 is equal or greater than the entry of the lookup table corresponding to the minority classes $\{t_2, \dots, t_l\}$. For problems with $l \leq 6$ the complete lookup table for all possible values of \mathbf{t} is computed. For problems with more than six classes, $k = 5$ is used to compute the lookup table. In these problems, if an instance receives votes for more than five classes after t queries, then the full ensemble is used to predict its class label. All the lookup tables are computed using $\pi = 99\%$ and $T = 101$.

Table 1 summarizes the results for the datasets investigated. The values reported correspond to averages over the 100 random partitions into training and test data. The standard deviations are also reported after the \pm symbols. The second column of the table displays l , the number of classes of each problem and, if different, the k value used to generate the table. If $k \neq l$, the percentage of instances that require querying all the networks in the ensemble is also indicated. The third column displays the average disagreement rates between the predictions of the full ensemble and the predictions of the ensemble when IB-pruning is used. The generalization error and standard deviation of these predictions are given in the fourth and fifth columns, respectively. Finally, the sixth column displays the average number of neural networks used by IB-pruning to classify each instance and the quartiles between parenthesis.

The disagreement rates between the pruned and the complete ensemble class estimates are under $1 - \pi$ in all the problems investigated. Furthermore, the differences in classification error between the complete ensemble and the IB-pruned

Table 2. Execution time

l	k	time (sec.)	C_{max}	table size
3	3	0.76	1,734	884
4	4	13.9	44,867	8,037
5	5	758.4	937,747	46,262
6	6	24,757	16,489,227	189,509
6	5	10,126	16,489,227	46,262
7	5	15,037	18,798,419	46,262
8	5	18,579	19,398,244	46,262
9	5	19,279	19,605,911	46,262
10	5	20,143	19,692,158	46,262
11	5	20,270	19,732,838	46,262
12	5	20,417	19,754,960	46,262

ensemble are almost negligible, well below the disagreement rates. This means that the changes in the class labels occur in approximately the same numbers of correctly and incorrectly classified instances. A paired two-sided Wilcoxon sign test at 5% indicates that these differences in classification error are not statistically significant in any of the problems investigated. The average number of neural networks used to classify the different instance varies significantly in different problems. This value is below 10 on average for *Iris*, *New-Thyroid* and *Wine*. For *Vowel* approximately half of the networks need to be queried on average. These figures indicate an increment of the classification speed in a factor that varies between 2 for *Vowel* and 10.7 for *Wine*. The smaller improvement for *Vowel* is due to the fact that 30.2% of instances have votes for more than 5 classes and, in consequence, they are classified using all the classifiers in the ensemble, i.e. 101 neural networks.

Table 2 displays the costs of constructing the lookup table for different values of l and k . The computations have been performed in a 2.4 Ghz Intel Xeon processor. The times reported in the second column of the table are given in seconds. The third column displays the values of C_{max} , the number of terms in the sum in (2) when $t = 0$. Finally, the last column gives the sizes of the lookup tables constructed. The time needed to compute the lookup table grows very quickly with l , when $k = l$ (first four rows of the table). This is mainly due to the fast increase in the number of terms in the sum in (2), and, to a lower extent, to the larger table size. If the value k is kept fixed (last seven rows of the table), the size of the table remains constant as l increases. In addition, the growth of the maximum number of terms in the sum in (2) is much slower. The combination of these factors leads to a fairly slow increment in the time needed to build the table for increasing values of l and constant k .

5 Conclusions

Majority voting is often used to combine the decisions of the classifiers that make up an ensemble. In most cases, it is not necessary to query all classifiers in the

ensemble to estimate the final class prediction with a high level of confidence. Assuming that, for a given instance, the predictions of the classifiers are independent random variables, it is possible to calculate the probability that the majority class obtained after querying t classifiers will not change after querying the remaining $T - t$ classifiers of the ensemble. If some uncertainty in the estimation of the class prediction of the complete ensemble is acceptable, the querying process can be stopped when this probability exceeds a specified threshold π , typically high, but smaller than 1. Because less classifiers are queried, this instance-based pruning procedure can be used to increase the classification speed, provided that the values of these probabilities are readily available. A possible solution is to compute them and then store them in a lookup table. The difficulty is that the size of the lookup table and the cost of computing the values to be stored in it grow very fast as the number class labels of the problem increases. In this article we propose several techniques that can be used to optimize this process, and allow the application of IB-pruning in multi-class problems. Experiments in problems with up to 11 classes show that the classification speed can be improved by a factor between ≈ 2 and ≈ 10 , depending on the problem considered, without any significant variation in the classification error of the ensemble. The optimized tables are smaller and can be stored in the working memory of a standard desktop computer. They can be computed off-line for a given ensemble size, number of classes (l) and value of k and can then be used for *any* classification task and ensemble type, provided that the classifiers are trained independently.

References

1. Breiman, L.: Bagging predictors. *Machine Learning* 24(2), 123–140 (1996)
2. Breiman, L.: Random forests. *Machine Learning* 45(1), 5–32 (2001)
3. Martínez-Muñoz, G., Suárez, A.: Switching class labels to generate classification ensembles. *Pattern Recognition* 38(10), 1483–1494 (2005)
4. Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(10), 1619–1630 (2006)
5. Martínez-Muñoz, G., Sánchez-Martínez, A., Hernández-Lobato, D., Suárez, A.: Class-switching neural network ensembles. *Neurocomputing* 71(13-15), 2521–2528 (2008); *Artificial Neural Networks (ICANN 2006)*
6. Hernández-Lobato, D., Martínez-Muñoz, G., Suárez, A.: Statistical instance-based pruning in ensembles of independent classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(2), 364–369 (2009)
7. Fan, W.: Systematic data selection to mine concept-drifting data streams. In: KDD 2004, pp. 128–137. ACM, New York (2004)
8. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
9. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S. Springer, New York (2002)

Robustness of Kernel Based Regression: A Comparison of Iterative Weighting Schemes

Kris De Brabanter¹, Kristiaan Pelckmans¹, Jos De Brabanter^{1,2},
Michiel Debruyne³, Johan A.K. Suykens¹, Mia Hubert⁴, and Bart De Moor¹

¹ KULeuven, ESAT-SCD, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
`{Kris.DeBrabanter,Kristiaan.Pelckmans,Johan.Suykens`
`Bart.DeMoor}@esat.kuleuven.be`

² KaHo Sint-Lieven (Associatie K.U.Leuven), Departement Ind. Ing., B-9000 Gent
`Jos.DeBrabanter@kahosl.be`

³ Universiteit Antwerpen, Department of Mathematics and Computer Science
Middelheimlaan 1G, B-2020 Antwerpen, Belgium
`Michiel.Debruyne@ua.ac.be`

⁴ KULeuven, Department of Statistics, Celestijnenlaan 200B,
B-3001 Leuven, Belgium
`Mia.Hubert@wis.kuleuven.be`

Abstract. It has been shown that Kernel Based Regression (KBR) with a least squares loss has some undesirable properties from robustness point of view. KBR with more robust loss functions, e.g. Huber or logistic losses, often give rise to more complicated computations. In this work the practical consequences of this sensitivity are explained, including the breakdown of Support Vector Machines (SVM) and weighted Least Squares Support Vector Machines (LS-SVM) for regression. In classical statistics, robustness is improved by reweighting the original estimate. We study the influence of reweighting the LS-SVM estimate using four different weight functions. Our results give practical guidelines in order to choose the weights, providing robustness and fast convergence. It turns out that Logistic and Myriad weights are suitable reweighting schemes when outliers are present in the data. In fact, the Myriad shows better performance over the others in the presence of extreme outliers (e.g. Cauchy distributed errors). These findings are then illustrated on toy example as well as on a real life data sets.

Keywords: Least Squares Support Vector Machines, Robustness, Kernel methods, Reweighting.

1 Introduction

Regression analysis is an important statistical tool routinely applied in most sciences. However, using least squares techniques there is an awareness of the dangers posed by the occurrence of outliers present in the data. Not only the response variable can be outlying, but also the explanatory part, leading to leverage points. Both types of outliers may totally spoil an ordinary LS analysis.

To cope with this problem, statistical techniques have been developed that are not so easily affected by outliers. These methods are called robust or resistant. A first attempt was done by Edgeworth [1]. He argued that outliers have a very large influence on LS because the residuals are squared. Therefore, he proposed the least absolute values regression estimator (L_1 regression). The second great step forward in this class of methods occurred in the 1960s and early 1970s with fundamental work of Tukey [2], Huber [3] and Hampel [4]. From their work the following methods were developed: M -estimators, Generalized M -estimators, R -estimators, L -estimators, S -estimators, repeated median estimator, least median of squares, Detailed information about these estimators as well as methods for robustness measuring can be found in [5], [6], [7] and [8].

All of the above mentioned techniques were originally proposed for parametric regression. In this paper we further investigate these ideas to the nonparametric case, more specifically for Least Squares Support Vector Machines (LS-SVM). Other recent work in this direction is [9], [10] and [11]. LS-SVMs were proposed by Suykens et al. [12] as a reformulation of the Support Vector Machines (SVM) [13], applicable to a wide range of problems in supervised and unsupervised learning. In case of LS-SVMs one works with equality instead of inequality constraints and a sum of squared error cost function is used. Due to this, the regression solution is found by solving a linear system instead of a convex quadratic programming problem. By using an L_2 cost function robustness properties are lost. A successful attempt to improve the robustness was given by Suykens et al. [14]. The technique is based on a two stage approach: first, classical LS-SVM is applied and secondly appropriate weighting values are computed taking the residuals of the first step into account. For LS-SVM this weighting technique can be employed cheaply and efficiently in order to robustify the solution. In this way the weighting procedure serves as an alternative to other robust estimation methods based on L_1 and Huber's loss function without giving rise to complicated computations.

In this paper we show that the weighted LS-SVM breaks down under non Gaussian noise distributions with heavy tails. In order to deal with these distributions a reweighting scheme is proposed. Different weight functions are investigated in order to compare their performance under these heavy tailed distributions. This paper is organized as follows. In Section 2 we briefly review the basic notions of weighted LS-SVM. Section 3 explains the practical difficulties associated with estimating a regression function when the data is contaminated with outliers. Section 4 describes some extensions of existing results in order to deal with outliers in nonparametric regression. The methods are illustrated on a toy example as well as on real life data sets in Section 5.

2 Weighted LS-SVM for Nonlinear Function Estimation

In order to obtain a robust estimate, one can replace the L_2 loss function in the LS-SVM formulation by e.g. L_1 or Huber's loss function. This would lead to a Quadratic Programming (QP) problem and hence increasing the computational

load. Instead of using robust cost functions, one can obtain a robust estimate based upon the previous LS-SVM solution. Given a training set defined as $\mathcal{D}_n = \{(X_k, Y_k) : X_k \in \mathbb{R}^d, Y_k \in \mathbb{R}; k = 1, \dots, n\}$ of size n drawn i.i.d. from an unknown distribution F_{XY} according to $Y_k = g(X_k) + e_k, \quad k = 1, \dots, n$, where $e_k \in \mathbb{R}$ are assumed to be i.i.d. random errors with $E[e_k | X = X_k] = 0$, $\text{Var}[e_k] = \sigma^2 < \infty$, $g \in C^z(\mathbb{R})$ with $z \geq 2$, is an unknown real-valued smooth function and $E[Y_k | X = X_k] = g(X_k)$. The optimization problem of finding the vector w and $b \in \mathbb{R}$ for regression can be formulated as follows [12]

$$\begin{aligned} \min_{w, b, e} \mathcal{J}(w, e) &= \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^n v_k e_k^2 \\ \text{s.t. } Y_k &= w^T \varphi(X_k) + b + e_k, \quad k = 1, \dots, n, \end{aligned} \quad (1)$$

where the error variables from the unweighted LS-SVM $\hat{e}_k = \hat{a}_k / \gamma$ (case $v_k = 1, \forall k$) are weighted by weighting factors v_k [14] according to (3) and $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^{n_h}$ is the feature map to the high dimensional feature space as in the standard (SVM) [13] case.

By using Lagrange multipliers, the solution of (1) can be obtained by taking the Karush-Kuhn-Tucker (KKT) conditions for optimality. The result is given by the following linear system [12] in the dual variables α

$$\left(\begin{array}{c|c} 0 & 1_n^T \\ \hline 1_n & \Omega + D_\gamma \end{array} \right) \left(\begin{array}{c} b \\ \alpha \end{array} \right) = \left(\begin{array}{c} 0 \\ Y \end{array} \right), \quad (2)$$

with $D_\gamma = \text{diag} \left\{ \frac{1}{\gamma v_1}, \dots, \frac{1}{\gamma v_n} \right\}$. The weights v_k are based upon $\hat{e}_k = \hat{a}_k / \gamma$ from the (unweighted) LS-SVM ($D_\gamma = I$). The weights v_k are given by [6]

$$v_k = \begin{cases} 1, & |\hat{e}_k / \hat{s}| \leq c_1; \\ \frac{c_2 - |\hat{e}_k / \hat{s}|}{c_2 - c_1}, & c_1 \leq |\hat{e}_k / \hat{s}| \leq c_2; \\ 10^{-8}, & \text{otherwise,} \end{cases} \quad (3)$$

where $\hat{s} = 1.483 \text{MAD}(\hat{e}_k)$ is a robust estimate of the standard deviation, where MAD is the Median Absolute Deviation. The constants are set to $c_1 = 2.5$ and $c_2 = 3$. Also $Y = (Y_1, \dots, Y_n)^T$, $1_n = (1, \dots, 1)^T$, $\alpha = (\alpha_1, \dots, \alpha_n)^T$ and $\Omega_{kl} = \varphi(X_k)^T \varphi(X_l) = K(X_k, X_l)$ for $k, l = 1, \dots, n$, with K a positive definite kernel. The resulting weighted LS-SVM model for function estimation becomes

$$\hat{g}(x) = \sum_{k=1}^n \hat{a}_k K(x, X_k) + \hat{b}. \quad (4)$$

3 Problems with Outliers in Nonparametric Regression

A number of problems, some quite fundamental, occur when nonparametric regression is attempted in the presence of outliers. In nonparametric regression, e.g. Nadaraya-Watson kernel estimates, local polynomial kernel estimates, spline estimates and wavelets estimates, the L_2 risk is often used. There are two reasons

for considering the L_2 risk: (i) this simplifies the mathematical treatment of the whole problem and (ii) trying to minimize the L_2 risk leads to estimates which can be computed rapidly. However, the L_2 risk can be very sensitive to regression outliers. A linear kernel (in kernel-based regression) leads to non-robust methods. On the other hand using decreasing kernels, i.e. kernels such that $K(u) \rightarrow 0$ as $u \rightarrow \infty$, leads to quite robust methods with respect to outliers in the x -space (leverage points). The influence for both $x \rightarrow \infty$ and $x \rightarrow -\infty$ is bounded in \mathbb{R} when using decreasing kernels. Common choices for decreasing kernels are: $K(u) = \max(1 - u^2, 0)$, $K(u) = \exp(-u^2)$ and $K(u) = \exp(-|u|)$.

This breakdown of kernel based nonparametric regression is illustrated by a simple simulated example in Figure I. Consider the following 200 observations $\{(X_1, Y_1), \dots, (X_{200}, Y_{200})\}$ according to the relation $f(X) = 1 - 6X + 36X^2 - 53X^3 + 22X^5$ and $X \sim U[0, 1]$. Two different types of outlier sets are added to the underlying function. The errors are normally distributed with variance $\sigma^2 = 0.05$ and $\sigma^2 = 0.1$ in Figure Ia. In Figure Ib and Figure Ic three outliers are added to the data. LS-SVM (unweighted case) cannot cope with the outliers showing a bump between 0.8 and 0.95. Notice that the unweighted LS-SVM only shows a local and not a global breakdown for the regression. SVM [3] on the other hand, deals with these type of outliers since it uses an ϵ -insensitive loss function. Figure Ic shows that the weighted LS-SVM method is able to handle these outliers and has a similar result as SVM. In Figure Id the distribution of the errors was given by the gross error model or ϵ -contamination model [3] and is defined as follows

$$\mathcal{U}(F_0, \epsilon) = \{F : F(e) = (1 - \epsilon)F_0(e) + \epsilon G(e), 0 \leq \epsilon \leq 1\} \quad (5)$$

where F_0 is some given distribution (the ideal nominal model), G is an arbitrary continuous distribution and ϵ is the first parameter of contamination. This contamination model describes the case, where with large probability $(1 - \epsilon)$, the data occurs with distribution F_0 and with small probability ϵ outliers occur according to distribution G . In this case the contamination distribution G was taken to be a cubic standard Cauchy distribution and $\epsilon = 0.3$. This distribution is quite special since its moments are not defined. Both robust methods fail to fit the underlying regression model.

4 Iteratively Reweighted Kernel Based Regression

In this Section we describe and compare four types of weight functions. Also convergence properties for each of the weight functions are given.

4.1 Weight Functions

Many weight functions have been proposed in literature, especially for linear regression [6]. Four of these weight functions $V : \mathbb{R} \rightarrow [0, 1]$, with $V(r) = \frac{\psi(r)}{r}$ satisfying $V(0) = 1$, are shown in Table I with corresponding loss function $L(r)$ and score function $\psi(r) = \frac{dL(r)}{dr}$. The first three weight function are quite common and are often used in regression [6][10]. The fourth function, Myriad with

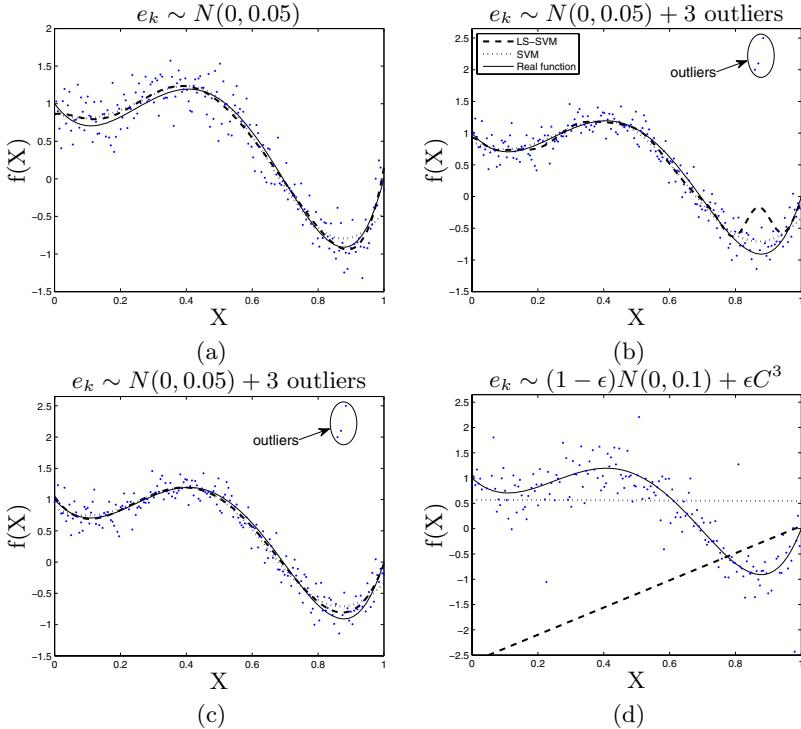


Fig. 1. Simulated data with two types of different outlier sets, fitted with LS-SVM (dashed line) and SVM (dotted line). The full line represents the underlying polynomial function. SVM and weighted LS-SVM (dashed line in (c) and (d)) can both handle the first type of outliers, but fail when the contamination distribution is taken to be a cubic standard Cauchy with $\epsilon = 0.3$. For visual reasons, not all data is displayed in (d).

parameter $\delta \in \mathbb{R}_0^+$, has been proposed in the area of statistical nonlinear signal processing [15]. The Myriad is derived from the Maximum Likelihood (ML) estimation of a Cauchy distribution [16] and is used as a robust location estimator in stable noise environments. When using the Myriad as a location estimator it can be shown that the Myriad offers a rich class of operation modes that can be controlled by varying the parameter δ . When the noise is Gaussian, large values of δ can provide the optimal performance associated with the sample mean, whereas for highly impulsive noise statistics, the resistance of mode-type estimators can be achieved by setting low values of δ . Arce [15] observed experimentally that values on the order of the data range, $\delta \approx X_{(n)} - X_{(1)}$, often make the Myriad an acceptable approximation to the sample average. We denote $X_{(m)}$ as the m -th order statistic for $m = 1, \dots, n$. Intermediate values of δ assume a sample set with some outliers and some well behaved samples. On the other side, when δ is small i.e. $\delta \approx \min_{i,j} |X_i - X_j|$, the Myriad is to be considered approximately a mode estimator.

Table 1. Definitions for the Huber, Hampel, Logistic and Myriad (with parameter $\delta \in \mathbb{R}_0^+$) weight functions $V(\cdot)$. The corresponding loss $L(\cdot)$ and score function $\psi(\cdot)$ are also given.

	Huber	Hampel	Logistic	Myriad
$V(r)$	$\begin{cases} 1, & \text{if } r < \beta; \\ \frac{\beta}{ r }, & \text{if } r \geq \beta. \end{cases}$	$\begin{cases} 1, & \text{if } r < b_1; \\ \frac{b_2 - r }{b_2 - b_1}, & \text{if } b_1 \leq r \leq b_2; \\ 0, & \text{if } r > b_2. \end{cases}$	$\frac{\tanh(r)}{r}$	$\frac{\delta^2}{\delta^2 + r^2}$
$\psi(r)$				
$L(r)$	$\begin{cases} r^2, & \text{if } r < \beta; \\ \beta r - \frac{1}{2}\beta^2, & \text{if } r \geq \beta. \end{cases}$	$\begin{cases} r^2, & \text{if } r < b_1; \\ \frac{b_2 r^2 - r ^3}{b_2 - b_1}, & \text{if } b_1 \leq r \leq b_2; \\ 0, & \text{if } r > b_2. \end{cases}$	$r \tanh(r)$	$\log(\delta^2 + r^2)$

One can obtain a robust estimate based upon the previous LS-SVM solutions using an iteratively reweighting approach. In the i -th iteration one can weight the error variables $\hat{e}_k^{(i)} = \hat{\alpha}_k^{(i)} / \gamma$ for $k = 1, \dots, n$ by weighting factors $v^{(i)} = (v_1^{(i)}, \dots, v_n^{(i)})^T \in \mathbb{R}^n$, determined by one of the four weighting functions in Table II. One obtains an iterative algorithm, see Algorithm II, to solve the problem.

Algorithm 1. Iteratively Reweighted LS-SVM

- 1: Given optimal learning parameters (γ, σ) , e.g. by cross-validation, and compute the residuals $\hat{e}_k = \hat{\alpha}_k / \gamma$ from the unweighted LS-SVM ($v_k = 1, \forall k$)
 - 2: **repeat**
 - 3: Compute $\hat{s} = 1.483 \text{MAD}(e_k^{(i)})$ from the $e_k^{(i)}$ distribution
 - 4: Determine the weights $v_k^{(i)}$ based upon $r^{(i)} = e_k^{(i)} / \hat{s}$ and the chosen weight function V in Table II
 - 5: Solve the weighted LS-SVM (2) with $D_\gamma = \text{diag} \left\{ \frac{1}{\gamma v_1^{(i)}}, \dots, \frac{1}{\gamma v_n^{(i)}} \right\}$, resulting the model $\hat{m}^{(i)}(x) = \sum_{k=1}^n \hat{\alpha}_k^{(i)} K(x, X_k) + \hat{b}^{(i)}$
 - 6: Set $i = i + 1$
 - 7: **until** consecutive estimates $\alpha_k^{(i-1)}$ and $\alpha_k^{(i)}$ are sufficiently close to each other $\forall k = 1, \dots, n$. In this paper we take $\max_k(|\alpha_k^{(i-1)} - \alpha_k^{(i)}|) \leq 10^{-4}$.
-

4.2 Speed of Convergence-Robustness Trade-Off

In a functional analysis setting it has been shown in [9] and [10] that the influence function [4] of reweighted Least Squares Kernel Based Regression (LS-KBR) with a bounded kernel converges to bounded influence function, even when the initial LS-KBR is not robust, if

- (c1) $\psi : \mathbb{R} \rightarrow \mathbb{R}$ is a measurable, real, odd function,
- (c2) ψ is continuous and differentiable,

- (c3) ψ is bounded,
- (c4) $E_{P_e} \psi'(e) > 0$ where P_e denotes the distribution of the errors. This condition can be relaxed into ψ is increasing.

The influence function (IF) describes the (approximate and standardized) effect of an additional observation in any point x on a statistic T , given a (large) sample with distribution F . Thus an unbounded IF means that an infinitesimal amount of outliers can have an arbitrary large effect.

Define

$$d = E_{P_e} \frac{\psi(e)}{e} \quad \text{and} \quad c = d - E_{P_e} \psi'(e), \quad (6)$$

then it can be shown [10] that c/d establishes an upper bound on the reduction of the influence function at each step. The upper bound represents a trade-off between the reduction of the influence function (speed of convergence) and the degree of robustness. The higher the ratio c/d the higher the degree of robustness but the slower the reduction of the influence function at each step and vice versa.

In Table 2 this upper bound is calculated at a Normal distribution, a standard Cauchy and a cubic standard Cauchy for the four types of weighting schemes. Note that the convergence of the influence function is quite fast, even at heavy tailed distributions.

For Huber and Myriad weights, the convergence rate decreases rapidly as β respectively δ increases. This behavior is to be expected, since the larger β respectively δ , the less points are downweighted. Also note that the upper bound on the convergence rate approaches 1 as $\beta, \delta \rightarrow 0$, indicating a high degree of robustness but slow convergence rate. A good choice between convergence and robustness is therefore Logistic weights. Also notice the small ratio for the Hampel weights indicating a low degree of robustness. The inability of these weights to handle extreme outliers is shown in the next Section. For further elaboration on the topic we refer the reader to [11].

5 Simulations

5.1 Toy Example

Recall the low order polynomial function in Section 3 with 200 observations according to $f(X) = 1 - 6X + 36X^2 - 53X^3 + 22X^5$ and $X \sim U[0, 1]$. The distribution of the errors is given by the gross error model, see Section 3, with $\epsilon = 0.3$, $F_0 = N(0, 0.1)$ and $G = C^3(0, 1)$. The results for the four types of weight functions are shown in Figure 2 and performances in the three norms are given in Table 3. For this simulation we set $\beta = 1.345$, $b_1 = 2.5$ and $b_2 = 3$ and $\delta = \frac{1}{2}[\hat{e}_{(\frac{3}{4}n)}^{(i)} - \hat{e}_{(\frac{1}{4}n)}^{(i)}]$ where $\hat{e}_{(m)}^{(i)}$ denotes the m -th order statistic of the residual \hat{e} in the i -th iteration. For all simulations, the learning parameters are tuned via 10-fold robust cross-validation. This simulation shows that the four weight functions are able to handle these extreme outliers. Although Hampel and Myriad weight functions do not satisfy the relaxed condition of (c4), condition (c4) is valid for common error distributions i.e. Normal, Cauchy, Student t , Laplace, This

Table 2. Values of the constants c , d and c/d for the Huber (with different cutoff values β), Logistic, Hampel and Myriad (for different parameters δ) weight function at a standard Normal distribution, a standard Cauchy and a cubic standard Cauchy. The bold values represent an upper bound for the reduction of the influence function at each step.

Weight function	Parameter settings	$N(0, 1)$			$C(0, 1)$			$C^3(0, 1)$		
		c	d	c/d	c	d	c/d	c	d	c/d
Huber	$\beta = 0.5$	0.32	0.71	0.46	0.26	0.55	0.47	0.0078	0.034	0.23
	$\beta = 1$	0.22	0.91	0.25	0.22	0.72	0.31	0.0022	0.037	0.059
	$\beta = 2$	0.04	0.99	0.04	0.14	0.85	0.17	0.0002	0.038	0.0053
Logistic		0.22	0.82	0.26	0.21	0.66	0.32	0.004	0.035	0.12
Hampel	$b_1 = 2.5$ $b_2 = 3$	0.006	0.99	0.006	0.02	0.78	0.025	0.00003	0.038	0.0007
Myriad	$\delta = 0.1$	0.11	0.12	0.92	0.083	0.091	0.91	0.007	0.009	0.83
	$\delta = 0.6475$	0.31	0.53	0.60	0.24	0.40	0.60	0.01	0.028	0.36
	$\delta = 1$	0.31	0.66	0.47	0.25	0.50	0.50	0.008	0.032	0.25

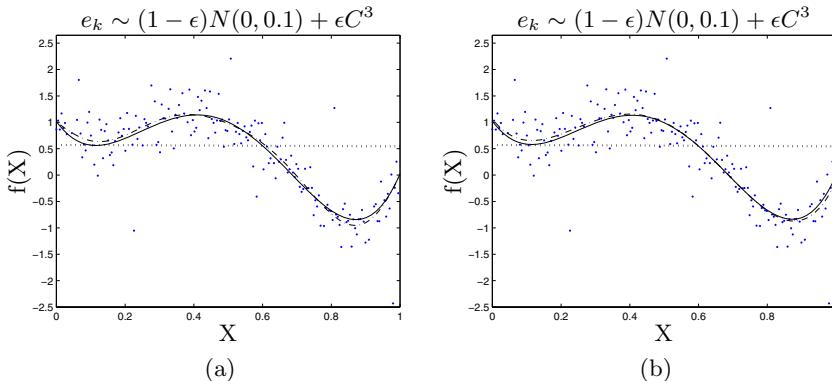


Fig. 2. Low order polynomial function with 200 observations according to $f(X) = 1 - 6X + 36X^2 - 53X^3 + 22X^5$ and $X \sim U[0, 1]$. The distribution of the errors is given by the gross error model with $\epsilon = 0.3$, $F_0 = N(0, 0.1)$ and $G = C^3(0, 1)$. The dotted line is the corresponding SVM fit. The iteratively reweighted LS-SVM with (a) Huber weights (full line) and Hampel weights (dash dotted line); (b) Logistic weights (full line) and Myriad weights (dash dotted line).

simulation shows the best performance for the Myriad weight function. This is to be expected since it was designed for such types of outliers.

5.2 Real Life Data Sets

The octane data [17] consist of NIR absorbance spectra over 226 wavelengths ranging from 1102 to 1552 nm. For each of the 39 production gasoline samples the octane number Y was measured. It is well known that the octane data set contains six outliers to which alcohol was added. Table 4 shows the result

Table 3. Performances in the three norms (difference between the estimated function and the true underlying function) of the different weight functions used in iteratively reweighted LS-SVM on the low order polynomial. The last column denotes the number of iterations i_{\max} needed to satisfy the stopping criterion in Algorithm 1

	L_1	L_2	L_∞	i_{\max}
Huber	0.06	0.005	0.12	7
Hampel	0.06	0.005	0.13	4
Logistic	0.06	0.005	0.11	11
Myriad	0.03	0.002	0.06	17

Table 4. Results on the Octane and Demographic data sets. For 200 simulations the medians and mean absolute deviations (between brackets) of three norms are given (on test data). i_{\max} denotes the number of iterations needed to satisfy the stopping criterion in Algorithm 1. The best results are bold faced.

		Octane				Demographic			
	weights	L_1	L_2	L_∞	i_{\max}	L_1	L_2	L_∞	i_{\max}
IRLS-SVM	Huber	0.19(0.03)	0.07(0.02)	0.51(0.10)	15	0.31(0.01)	0.14(0.02)	0.83(0.06)	8
	Hampel	0.22(0.03)	0.07(0.03)	0.55(0.14)	2	0.33(0.01)	0.18(0.04)	0.97(0.02)	3
	Logistic	0.20(0.03)	0.06(0.02)	0.51(0.10)	18	0.30(0.02)	0.13(0.01)	0.80(0.07)	10
	Myriad	0.20(0.03)	0.06(0.02)	0.50(0.09)	22	0.30(0.01)	0.13(0.01)	0.79(0.06)	12
WLS-SVM		0.22(0.03)	0.08(0.02)	0.60(0.15)	1	0.33(0.02)	0.15(0.01)	0.80(0.02)	1
	SVM				-	0.37(0.02)	0.21(0.02)	0.90(0.06)	-

(medians and mean absolute deviations) of a Monte Carlo simulation (200 times) of the iteratively reweighted LS-SVM (IRLS-SVM), weighted LS-SVM (WLS-SVM) and SVM in different norms on a randomly chosen test set of size 10. As a next example consider the data about the demographical information on the 50 states of the USA in 1980. The data set provides information on 25 variables. The goal is to determine the murder rate per 100,000 population. The result is shown in Table 4 for randomly chosen test sets of size 15. The results of the simulations show that by using reweighting schemes the performance can be improved over weighted LS-SVM and SVM. To illustrate the trade-off between the degree of robustness and speed of convergence, the number of iterations i_{\max} are also given in Table 4. The stopping criterion was taken identically to the one in Algorithm 1. The number of iterations, needed by each weight function, confirms the results in Table 2.

6 Conclusion

In this paper we have compared four different type of weight functions and their use in iterative reweighted LS-SVM. We have shown through simulations that reweighting is useful when outliers are present in the data. By using an upper

bound for the reduction of the influence function we have demonstrated the existence of a trade-off between speed of convergence and the degree of robustness. The Myriad weight function is highly robust against (extreme) outliers but has a slow speed of convergence. A good compromise between speed of convergence and robustness can be achieved by using Logistic weights.

Acknowledgements. Research supported by: Research Council KUL: GOA AMBioRICS, CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects G.0452.04, G.0499.04, G.0211.05, G.0226.06 1, G.0321.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, research communities (ICCoS, ANMMM, MLDM); IWT: PhD Grants, McKnow-E, Eureka-Flite+; Helmholtz: vicERP. Belgian Federal Science Policy Office: IUAP P6/04 (DYSKO, 2007-2011); EU: ERNSI.

References

1. Edgeworth, F.Y.: On Observations Relating to Several Quantities. *Hermathena* 6, 279–285 (1887)
2. Tukey, J.W.: A survey of sampling from contaminated distributions. In: Olkin, I. (ed.) *Contributions to Probability and Statistics*, pp. 448–485. Stanford University Press, Stanford (1960)
3. Huber, P.J.: Robust Estimation of a Location Parameter. *Ann. Math. Stat.* 35, 73–101 (1964)
4. Hampel, F.R.: A General Definition of Qualitative Robustness. *Ann. Math. Stat.* 42, 1887–1896 (1971)
5. Huber, P.J.: *Robust Statistics*. Wiley, Chichester (1981)
6. Rousseeuw, P.J., Leroy, A.M.: *Robust Regression and Outlier Detection*. Wiley, Chichester (2003)
7. Maronna, R., Martin, D., Yohai, V.: *Robust Statistics*. Wiley, Chichester (2006)
8. Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A.: *Robust Statistics: The Approach Based on Influence Functions*. Wiley, Chichester (1986)
9. Christmann, A., Steinwart, I.: Consistency and Robustness of Kernel Based Regression in Convex Risk Minimization. *Bernoulli* 13(3), 799–819 (2007)
10. Debruyne, M., Christmann, A., Hubert, M., Suykens, J.A.K.: Robustness and Stability of Reweighted Kernel Based Regression. Technical Report 06-09, Department of Mathematics, K.U.Leuven, Leuven, Belgium (2008)
11. Debruyne, M., Hubert, M., Suykens, J.A.K.: Model Selection in Kernel Based Regression using the Influence Function. *J. Mach. Learn. Res.* 9, 2377–2400 (2008)
12. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: *Least Squares Support Vector Machines*. World Scientific, Singapore (2002)
13. Vapnik, V.N.: *Statistical Learning Theory*. Wiley, Chichester (1999)
14. Suykens, J.A.K., De Brabanter, J., Lukas, L., Vandewalle, J.: Weighted Least Squares Support Vector Machines: Robustness and Sparse Approximation. *Neurocomputing* 48(1-4), 85–105 (2002)

15. Arce, G.R.: Nonlinear Signal Processing: A Statistical Approach. Wiley, Chichester (2005)
16. Gonzalez, J.G., Arce, G.R.: Weighted Myriad Filters: A Robust Filtering Framework derived from Alpha-Stable Distributions. In: Proceedings of the 1996 IEEE Conference one Acoustics (1996)
17. Hubert, M., Rousseeuw, P.J., Vanden Branden, K.: ROBPCA: a New Approach to Robust Principal Components Analysis. *Technometrics* 47, 64–79 (2005)

Mixing Different Search Biases in Evolutionary Learning Algorithms

Kristina Davoian and Wolfram-M. Lippe

Department of Mathematics and Computer Science,
University of Münster, Einsteinstr. 62, 48149 Münster, Germany
{kristina.davoian,lippe}@uni-muenster.de

Abstract. This work investigates the benefits of using different distribution functions in the evolutionary learning algorithms with respect to Artificial Neural Networks' (ANNs) generalization ability. We examine two modification of the recently proposed network weight-based evolutionary algorithm (NWEA), by mixing mutation strategies based on three distribution functions at the chromosome and the gene levels. The utilization of combined search strategies in the ANNs training implies that different step sizes determined by mixed distributions will direct the evolution towards good generalized ANNs.

Keywords: Artificial Neural Networks, Learning, Evolutionary Algorithms.

1 Introduction

Evolutionary Algorithms (EAs) have found wide application in optimization of Artificial Neural Networks parameters, since they outperform the originally proposed gradient-descent learning approaches in terms of computations speed, simplicity and resistance to local minima trapping. However, mutation-based EAs, i.e. Evolutionary Programming (EP) and Evolutionary Strategies (ES) have proven to be more efficient in ANNs' learning than Genetic Algorithms, which due to their primary search operator (crossover) often face the permutation problem [1], [2], [14].

The key aspects that EP and ES concentrate on are the self-adaptive methods for changing the strategy parameters and the distribution used in mutation. The classical EP and ES algorithms utilize the standard normal distribution and similar self-adaptive methods, introduced by Rechenberg [3] and Schwefel [4] for ES and independently, by Fogel [5], [6] for meta-EP (widely known as classical EP). Later, Yao et al. [7] established that the distribution in the mutation strategy is crucial in the determination of the mutation step size, and proposed a novel EP technique, called the Fast Evolutionary Programming (FEP) [8], [9], which adopts the self-adaptation strategy of the classical EP (CEP), but uses the Cauchy distribution instead of the Gaussian one. Further, Yao investigated the impact of utilizing both Gaussian and Cauchy distributions at the chromosome and gene level and introduced modifications of FEP, referred to

as the improved Fast Evolutionary Programming (IFEP) and the mixed Fast Evolutionary Programming (MEP), respectively [13].

This work is concerned with the improvement of the evolutionary learning in ANNs. The goal for this work is to investigate the impact of mixing search biases of mutations based on three different distributions, on the generalization in ANNs. More specifically, we study how different step sizes determined by mixed distributions improve the quality affect the algorithm's convergence speed and the quality of evolved ANNs. We introduce two modifications of the recently proposed network weight (NW)-based EA (NWEA) strategy [10], which combine Gaussian, Cauchy and uniform distributions at the chromosome and gene levels in the learning strategy (the original NWEA strategy is based on the uniform distribution). In contrast to the CEP and FEP approaches, which are independent search methods, the NWEA algorithm was developed specially for ANNs' learning. The main feature of NWEA consists in the special approach to the adjustment of the random values. The strategy parameter in the classical techniques is evolved during the evolution alongside with the object parameters. In comparison to that, the adaptation strategy in NWEA consists of two components, which bear the information about the position of an individual in the search space and based of this knowledge, bias the improvement towards the perspective regions of the search space. The first modification of NWEA, referred further to as combined NWEA (CNWEA) produces three offspring as a result of mutation by using Gaussian, Cauchy and uniform random values, respectively (i.e. provides mutation at the chromosome level). The second modification uses three distributions to generate one offspring (i.e. carries out mutation at the gene level). The utilization of particular type of distribution is defined by a certain probability.

In order to evaluate the ANNs produced by CNWEA and MNWEA, the preliminary experimental studies on the breast cancer and heart disease diagnosis data sets were provided. The generalization results of ANNs, evolved by CNWEA and MNWEA were compared with those, evolved by NWEA, which is shown to be more efficient than classical searching techniques [11].

The rest of the paper is organized as follows: Section 2 discusses the advantages of using Gaussian and Cauchy distributions in terms of step size. Section 3 described the features of the NWEA adaptation strategy and main steps of the learning algorithm. Sections 4 and 5 introduce the combined NWEA (CNWEA) and the mixed NWEA (MNWEA), respectively. Following that, Section 6 presents the experiments and analyses the obtained results. Finally, Section 7 concludes this paper.

2 Length of Gaussian and Cauchy Jumps

Both CEP and FEP use the same mutation scheme to modify individuals and the same self-adaptation strategy to correct mutation step size. The only difference is the distribution used to generate random numbers. Therefore, it is reasonable to claim that the features of algorithms' performances are caused by a type of used distribution. Let us discuss the advantages of using both distributions.

The expected length of Gaussian (with $\mu = 0$ and $\sigma^2 = 1$) and Cauchy (with $\gamma = 1$) jumps can be calculated by integrating their probability density functions:

$$E_G(x) = \int_0^{+\infty} x \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \frac{1}{\sqrt{2\pi}} = 0.399$$

$$E_C(x) = \int_0^{+\infty} x \frac{1}{\pi(1+x^2)} dx = +\infty$$

Apparently, the Cauchy distribution enables longer jumps than the Gaussian one. At first sight it seems that longer jumps in the search space induce quicker convergence, and so the Cauchy distribution is preferable in the searching strategy. However, this assumption is wrong. The analytical studies in [3] provided to investigate when large jumps are beneficial, showed that long jumps are advantageous only when the global optimum is far away from the current search point. In other words, long jumps are effective when the distance between the global optimum and the current point is larger than mutation's step size. On the other hand, the Cauchy distribution will no longer be beneficial when the distance between the neighbourhood of the global optimum and the current point is smaller than the step size of the mutation. This implies that the use of small jumps is more effective near the neighbourhood of the global optimum. Hence, the Gaussian distribution increases the probability of finding the optimum when the distance between the current point and the neighbourhood of the global optimum is small.

3 Network Weight-Based Evolutionary Algorithm (NWEA)

The basic step of the self-adaptation mechanism in CEP and FEP consists of *a mutation of mutation parameters themselves*, i.e. the evolution of strategy parameters alongside with the object parameters. The modification of the control parameters is realized by multiplication with a random variable.

In contrast to the classical evolutionary algorithms, the NWEA algorithm [10], designed to evolve ANNs' parameters, uses different self-adaptation approach to find an optimum. The self-adaptation in NWEA comprises two control parameters, which *incorporate genotype and phenotype information about the position of an individual in search space*. The first component includes information about worth of a chromosome according to its fitness. The second component adds information about the current ANN topology, the genotype encodes, i.e. information about position of an individual in the ANN architecture space. Alike ES and EP, the NWEA approach relies on mutation and does not utilize crossover at all.

The evolution with the NWEA algorithm is implemented as follows:

1. Create an initial population consisting of randomly generated chromosomes. Each chromosome $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(k)})$, $\forall i \in \{1, \dots, \mu\}$, represents one

possible set of connection weights¹, where l is a population size, k is a total number of connections between neurons and $x_i^{(j)} \in [-1.0; 1.0]$, $j \in \{1, \dots, k\}$ is a connection weight.

2. Evaluate the fitness of each individual from a population according to the objective function.
3. In contrast to other EP approaches, which apply probabilistic selection methods for choosing parents for reproduction, NWEA "allows" all parental chromosomes to take part in the creation of new individuals. It is worth noting that during mutation only one gene in the parental chromosome changes its value. Offspring chromosomes are created by application of the following equation to every chromosome in the population:

$$x_i^{(j)'} = x_i^{(j)} (1.0 + N_W(l, \bar{n}) \cdot N_E \cdot N_{Rand}^D), \quad (1)$$

where $x_i^{(j)}$ is a gene randomly chosen out of a chromosome x_i and mutated, $N_W(l, \bar{n})$ is a value, called *network weight*, that implicitly describes an ANN's internal structure, N_E represents an error, determined by the error function (MSE or other) of x_i , and N_{Rand}^D is a uniformly distributed random value.

The components N_E and $N_W(l, \bar{n})$ represent the adjustment components in the mutation strategy. They add knowledge about position and worth of a chromosome in search space in order to achieve the optimal improvement of chromosomes at each stage of evolution. The component N_E in Eq. (1) represents the genotype information, i.e. error of the mutated chromosome, which changes dynamically for every mutated chromosome. It enables the control of a randomly generated value and the adjustment of the mutation strength to an individual depending on its fitness, i.e. the higher the error of a chromosome, the higher the step size.

The value $N_W(l, \bar{n})$ depends on the number of hidden layers l and the average number of neurons in hidden layers \bar{n} in the given ANN and is defined by Eq. (2). This value is distributed by the Fermi-Dirac-like function and is calculated according to the following formula:

$$N_w(l, \bar{n}) = A_1 + \frac{l}{2} + \frac{B_1 - \frac{l}{2}}{1 + \exp\left(\frac{\bar{n} - \mu}{T_1}\right)} \quad (2)$$

The value μ is similar to the chemical potential in the original Fermi-Dirac function (as cited in [10]) and depends on the number of hidden layers:

$$\mu = A_2 + \frac{B_2}{1 + \exp\left(\frac{l - B_2}{T_2}\right)} \quad (3)$$

The coefficients $A_1 = 3.0$, $B_1 = 2.0$, $T_1 = 0.4$, $A_2 = 1.2$, $B_2 = 3.2$, $T_2 = 0.6$ were obtained by the approximation of the results in [10] so that the N_W values never become negative (the coefficients T_1 and T_2 correspond to

¹ In case of the simultaneous evolution of ANN's weights and architectures, each chromosome consists of a set of vectors according to the ANN's connectivity matrix.

the temperature in the original Fermi-Dirac function). For each ANN the quantity N_W is calculated only once and does not change its value during the evolution, if we consider the evolution of connection weights in the environment determined by an ANN architecture; in case of simultaneous evolution of architectures and connection weights it becomes a new value every time when ANN's architecture is changed.

The main advantage of incorporating phenotype information in mutation is that it comprises detailed knowledge about the ANN's topology² and thus enables to improve the values of connection weights in respect to a given phenotype. It is known from the theory of evolution that individuals with favorable traits, determined by the genotype, are more likely to survive and reproduce ("survival of the fittest") and the fitness of every individual is defined by the individual's ability to adapt to the environment. From such point of view, the NWEA approach is an abstraction, which involves the knowledge of an environment (phenotype) and adapts genotype of every individual to it, and thus, increases the fitness of chromosomes of every next generation.

4. Evaluate the fitness of a new individual based on the objective function.
5. Repeat the process from point (3) until λ ($\lambda \geq \mu$) new chromosomes are created.
6. Create new population of m individuals: new population is created according to the $(\mu + \lambda)$ -ES elitist method, which chooses μ best individuals from both parental and offspring chromosomes based on their fitness. This is accomplished by applying 2-tournament selection method that selects a group of individuals (usually four) from both parental and offspring populations, and compares their fitness. The individual with the higher fitness reaches the offspring population.
7. Repeat the process from point (2) until some halting criteria are satisfied.

Thus, by creating offspring population our greedy modification of the NWEA strategy selects the current best individuals. On the other hand, the risk of trapping in local optima is minimal, since the random values initially have long and short step sizes.

4 Combined NWEA (CNWEA)

The main idea behind combined NWEA (CNWEA) is to mix different search biases of mutations utilizing Gaussian, Cauchy and uniform distributions, at the chromosome level. The benefits of using Gaussian and Cauchy distributions were described in section 3. The utilization of the uniformly distributed random values does not have strong motivation; however it adds an additional randomness in the evolution process.

² Although mutation strategy in (1) incorporates the knowledge about ANN's internal structure, it gives detailed information about considering topology, since the number of neurons in input and output layers is determined by a solving problem.

The implementation of CNWEA is simple and differs from NWEA only in point 3 of the algorithm described in section 4.2. Each parental chromosome undergo mutation is modified three times and thus, produces three different offspring by using different values of N_{Rand}^D in Eq. (1): the first offspring is created using normally distributed values $N_{Rand}^{D_G}$ with mean $\mu = 0$ and variance $\sigma^2 = 1$, i.e.

$$x_i^{(j)'} = x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N_{Rand}^{D_G} \right), \quad (4)$$

the second offspring – by using Cauchy random numbers $N_{Rand}^{D_C}$ with a scale parameter $\gamma = 1$, i.e.

$$x_i^{(j)'} = x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N_{Rand}^{D_C} \right), \quad (5)$$

and the third offspring – by utilizing uniformly distributed random values, $N_{Rand}^{D_U} \in [-1.0, 1.0]$:

$$x_i^{(j)'} = x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N_{Rand}^{D_U} \right) \quad (6)$$

The best offspring is selected as a survivor. The rest of CNWEA is the same as NWEA.

5 Mixed NWEA (MNWEA)

An alternative way to mixing different biases is to combine mutation operators, based on Gaussian, Cauchy and uniform distributions at the gene rather than chromosome level. In our second modification of NWEA, called mixed NWEA (MNWEA) we define certain probabilities to apply Gaussian, Cauchy or uniform random numbers in the mutation strategy. Thus, some genes in the chromosome will be modified with the probability p_G according to Eq. (4), others will be mutated with the probability p_C according to Eq. (5) and rest – with the probability p_U according to Eq. (6).

$$x_i^{(j)'} = \begin{cases} x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N_{Rand}^{D_G} \right), & \text{with } p_G \\ x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N_{Rand}^{D_C} \right), & \text{with } p_C \\ x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N_{Rand}^{D_U} \right), & \text{with } p_U \end{cases}$$

where p_G , p_C and p_U are the probabilities of applying mutations according to the Eq.(4), Eq. (5) and Eq. (6), respectively, and $p_G + p_C + p_U = 1$. In our experiments we set the values p_G , p_C and p_U to 0.4, 0.4 and 0.2, respectively, as we aim at exploring the impact of small and large jumps provided by Gaussian and Cauchy distributions (see Section 2).

6 Experiments

The experimental studies of CNWEA and MNWEA were provided for breast cancer and heart disease diagnosis for the purpose of studying generalization ability of ANNs. In order to reduce the noise in the fitness evaluation, the ANNs' weights and architectures have been optimized simultaneously during the evolution. For each problem 50 runs of both algorithms were provided. Following initial parameters were used for these experiments: the population size 30 for MNWEA and 10 for CNWEA (since CNWEA produces three offspring from each parent), the maximum number of generations 300, and the number of hidden nodes for each individual was chosen uniformly at random between 1 and 3. The algorithms stopped when the maximal generation was reached.

6.1 Breast Cancer Diagnosis

The breast cancer data set was originally obtained from Dr. William H. Wolberg at the University of Wisconsin Hospitals, Madison. The data set consists of 699 examples of which 458 (65.5%) are benign examples and 241 (34.5%) are malignant examples. Each example contains nine attributes: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, mitoses. The goal of the data set is to classify a tumour as either benign or malignant based on these attributes.

In our experiments, the whole data set was divided into three subsets, as suggested by Prechelt (1994): a training set, a validation set, and a testing set. The first set was used to train EANNs. The validation set was explored as a pseudo-testing set in order to evaluate the fitness of networks during evolution. This prevents overtraining of the network and improves its generalization ability. During this process ANN's learning is carried out until the minimal error on the validation set (and not on the training set) is achieved. Finally, the testing data were considered to evaluate the performance of the evolved ANNs. 349 examples of the given breast cancer data set were used as training data, the following 175 examples as validation data, and the final 175 patterns as training data.

The error function (fitness) was calculated according to the equation, proposed by Prechelt [12]:

$$E = 100 \cdot \frac{o_{\max} - o_{\min}}{N \cdot P} \sum_{p=1}^P \sum_{i=1}^N (o_{pi} - t_{pi})^2$$

where o_{\min} and o_{\max} are the minimum and maximum values of output coefficients in the problem representation. N is the number of output nodes, P is the number of patterns, o_{pi} and t_{pi} are the actual and desired outputs of node i for pattern p correspondingly.

Table 1 shows the architecture of evolved ANNs as well as generation numbers (min and mean) at which the optimal results were obtained. Table 2 presents the classification results for breast cancer diagnosis. The value "rate" in the Table 2 shows the percentage of incorrect classified samples.

Table 1. ANN architectures for breast cancer diagnosis

	NWEA	CNWEA	MNWEA
Connections (min)	14	14	14
Connections (max)	78	82	86
Connections (mean)	36	29	36
Hidden nodes (min)	0	0	0
Hidden nodes (max)	4	4	4
Hidden nodes (mean)	1.3	1.3	1.4
Generation (min)	101	87	104
Generation (mean)	139.1	118	123.7

Table 2. Comparative results of the prediction accuracy for breast cancer diagnosis

	NWEA			CWEA			MNWEA		
	min	max	mean	min	max	mean	min	max	mean
Error, training	1.418	3.650	2.722	1.183	2.659	2.329	1.377	3.247	2.798
Rate, training	0.01698	0.04672	0.03921	0.00744	0.02451	0.02246	0.00954	0.0313	0.03422
Error, validation	0.052	1.018	0.557	0.037	0.637	0.349	0.034	0.924	0.503
Rate, validation	0.00000	0.01072	0.00547	0.00000	0.00902	0.00562	0.00000	0.01091	0.00536
Error, testing	0.178	3.546	1.413	0.054	2.899	1.217	0.117	3.487	1.406
Rate, testing	0.00000	0.03397	0.01384	0.00000	0.02687	0.00467	0.00000	0.03455	0.01424

6.2 Heart Disease Diagnosis

The heart disease data set was obtained from Cleveland Clinic Foundation and was supplied by Robert Detrano of the V.A. Medical Center, Long Beach, CA. The data set consists of 270 examples. The heart disease original data set consisted of 303 examples, but 6 of them contained missing class values and were excluded from the database. Other 27 examples of the remained data were eliminated as they retained in case of dispute.

Each example in the database contains 13 attributes, which present results of medical tests provided on patients: age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar < 120 (true or false), resting electrocardiogram (norm, abnormal or hyper), max heart rate, exercise induced angina, oldpeak, slope, number of vessels colored and thal (normal, fixed, rever). These attributes have been extracted from a larger set of 75. The goal of diagnosis is to recognize the presence or absence of heart disease given the attributes. Initially, the data set considered four different degrees of the heart disease to classify the predicted results. Later, modification in the problem definition suggested reducing the number of predicted values on two and categorizing results into two classes: presence or absence of illness.

For this set of experiments we applied the same equation to calculate fitness values as for the breast cancer diagnosis problem. Table 3 presents the ANN architectures for heart disease problem. Table 4 reports the generalization accuracy of ANNs, evolved by NWEA, CNWEA and MNWEA.

Table 3. ANN architectures for heart disease diagnosis

	NWEA	CNWEA	MNWEA
Connections (min)	28	26	26
Connections (max)	202	192	200
Connections (mean)	88.4	78.3	82,6
Hidden nodes (min)	2	2	2
Hidden nodes (max)	8	8	8
Hidden nodes (mean)	4.3	3.7	4.1
Generation (min)	127	106	113
Generation (mean)	172.2	157.9	169.6

Table 4. Comparative results of the prediction accuracy for heart disease diagnosis

	NWEA			CNWEA			MNWEA		
	min	max	mean	min	max	mean	min	max	mean
Error, training	7.489	12.166	11.007	5.763	12.005	8.963	5.893	12.137	7.774
Rate, training	0.07879	0.15184	0.12477	0.04831	0.12069	0.09446	0.05271	0.15040	0.10645
Error, validation	11.746	14.301	12.450	9.271	12.158	9.677	9.814	12.816	10.240
Rate, validation	0.12124	0.19706	0.15935	0.08113	0.13812	0.10543	0.09276	0.13762	0.12418
Error, testing	10.126	13.842	12.266	7.009	12.932	10.633	7.112	13.004	10.458
Rate, testing	0.13195	0.17997	0.15165	0.09385	0.14889	0.11676	0.11243	0.17002	0.12276

7 Conclusions

In this paper we have investigated mixing mutation strategies based on different distributions and proposed two modifications of the NWEA learning strategy for ANNs training. A combined NWEA (CNWEA) uses mutation strategies based on Gaussian, Cauchy and uniform distributions at the chromosome level. The mixed NWEA (MNWEA) uses the same mutation strategies as CNWEA, but combines them at the gene level.

The evolution process has been observed under different step sizes determined by mixed distributions. We have compared generalization accuracy of ANNs using the suggested mutation strategies on two simple benchmark data sets. According to our preliminary experiments, both CNWEA and MNWEA evolved compact ANNs with high training and generalization accuracy. The difference in evolved ANN architectures on both modifications was insignificant compared to NWEA; however both of them demonstrated higher generalization accuracy, especially CNWEA. Statistical analysis of data reported in Tables 2 and 4 with t-test showed that the differences between the error accuracies on both training and testing sets are not significant for breast cancer diagnosis (Table 2) and extremely significant for heart disease problem (Table 4). Both modifications have demonstrated higher convergence speed compared to NWEA as measured by the number of iterations before an optimal network is obtained. The preliminary results obtained for CNEWA and MNWEA are promising and encourage further studies on data sets with large number of attributes.

References

1. Yao, X.: A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems* 8(4), 539–567 (1993)
2. Yao, X.: Evolving artificial neural networks. In: *Proceedings of the IEEE*, pp. 1423–1447. IEEE Press, Los Alamitos (1999)
3. Rechenberg, I.: Cybernetic solution path of an experimental problem. In: Royal Aircraft Establishment, Farnborough, page Library Translation 1122 (1965)
4. Schwefel, H.-P.: *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Diplomarbeit, Technische Universität Berlin (1965)
5. Fogel, L.J.: Autonomous automata. *Industrial Research* 4, 14–19 (1962)
6. Fogel, L.J., Owens, A.J., Walsh, M.J.: *Artificial intelligence through simulated evolution*. Wiley, New York (1966)
7. Yao, X., Liu, Y.: An Analysis of evolutionary algorithms based on neighborhood and step size. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 297–307. Springer, Heidelberg (1997)
8. Yao, X., Liu, Y.: Fast Evolutionary Programming. In: Proc. of the Fifth Annual Conference on Evolutionary Programming, pp. 451–460. MIT Press, Cambridge (1996)
9. Yao, X., Liu, Y.: Evolutionary programming made faster. In: *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102. IEEE Press, Los Alamitos (1999)
10. Davoian, K., Lippe, W.-M.: Including phenotype information in mutation to evolve artificial neural networks. In: Proc. of the IEEE International Joint Conference on Neural Networks (IJCNN 2007), Orlando, USA (2007)
11. Davoian, K., Lippe, W.-M.: Exploring the role of activation function type in evolutionary artificial neural networks. In: Proc. of the 2008 Int. Conference on Data Mining (DMIN 2008), pp. 443–449. CSREA Press, Las Vegas (2008)
12. Prechelt, L.: Proben1-A set of neural network benchmark problems and benchmarking rules. Fakultät für Informatik, Universät Karlsruhe, Germany, Tech. Rep. 21/94 (1994)
13. Yao, X., Liu, Y.: Scaling up evolutionary programming algorithms. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 103–112. Springer, Heidelberg (1998)
14. Lippe, W.-M.: *Soft-Computing mit Neuronalen Netzen, Fuzzy-Logic und Evolutionären Algorithmen*. Springer, Heidelberg (2006)

Semi-supervised Learning for Regression with Co-training by Committee

Mohamed Farouk Abdel Hady*, Friedhelm Schwenker, and Günther Palm

Institute of Neural Information Processing
University of Ulm
D-89069 Ulm, Germany

{mohamed.abdel-hady,friedhelm.schwenker,guenther.palm}@uni-ulm.de

Abstract. Semi-supervised learning is a paradigm that exploits the unlabeled data in addition to the labeled data to improve the generalization error of a supervised learning algorithm. Although in real-world applications regression is as important as classification, most of the research in semi-supervised learning concentrates on classification. In particular, although Co-Training is a popular semi-supervised learning algorithm, there is not much work to develop new Co-Training style algorithms for semi-supervised regression. In this paper, a semi-supervised regression framework, denoted by *CoBCReg* is proposed, in which an ensemble of diverse regressors is used for semi-supervised learning that requires neither redundant independent views nor different base learning algorithms. Experimental results show that *CoBCReg* can effectively exploit unlabeled data to improve the regression estimates.

1 Introduction

Many real-world data mining applications have a large amount of unlabeled data but labeling data is often difficult, expensive, or time consuming, as it requires the effort of human experts for annotation. *Semi-supervised learning (SSL)* refers to methods that exploits the unlabeled data in addition to the labeled data to improve the generalization error of a supervised learning algorithm. Readers interested in recent advances of *SSL* are directed to [1].

Co-Training is a popular *SSL* paradigm introduced by Blum and Mitchell [2] where two classifiers are trained iteratively on two sufficient and independent views. That is, two sets of features that are conditionally independent given the class and each of which is sufficient for learning. At the initial iteration, two classifiers are trained using the available labeled training examples. Then at each further iteration, each classifier labels and selects some unlabeled examples

* This paper is based on work done within the Transregional Collaborative Research Centre SFB/TRR 62 *Companion-Technology for Cognitive Technical Systems* funded by the German Research Foundation (DFG). The first author was supported by a scholarship of the German Academic Exchange Service (DAAD) and a travel grant from the European Neural Network Society (ENNS).

to augment the training set of the other. The aim is that one classifier can improve the accuracy of the other by providing it with informative examples. Although, multi-view *Co-Training* is applicable on certain domains, its multi-view requirement is impractical in many real-world applications. Goldman and Zhou [3] presented a single-view *SSL* method, called *Statistical Co-learning*. Two different supervised learning algorithms are used to partition the input space into a set of equivalence classes and k -fold cross validation is applied:(1) to select the most confident examples to label at each iteration and (2) to combine the two hypotheses producing the final decision. Zhou and Li [4] present a new *Co-Training* style *SSL* method called *Tri-Training*. An initial ensemble of three classifiers is trained by data sets generated via bootstrap sampling from the original labeled training set [5]. These classifiers are then refined during the *Tri-Training* process, and the final hypothesis is produced via majority voting.

Although the success of the above *SSL* approaches for classification, there is not much work on *SSL* for regression. Zhou et al. [6] proposed a *Co-Training* style semi-supervised regression algorithm called *CoReg*. This algorithm employs two diverse k-Nearest Neighbor (kNN) regressors that were instantiated using two different values of the Minkowski distance order. The *labeling confidence* is estimated such that the most confidently labeled example is the one which keeps the regressor most consistent with the existing labeled training set.

Our main contributions are: (1) A new single-view committee-based semi-supervised regression algorithm, called *CoBCReg* that extends the standard *Co-Training* algorithm. It is based on an ensemble of *RBF network* regressors constructed by *Bagging* [5]. (2) A new *Gaussian basis function* that is based on Minkowski distance instead of Euclidean distance. For the effectiveness of *CoBCReg*, there must be some diversity among the committee members and *CoBCReg* should maintain this diversity during the *SSL* process. This is achieved not only by training regressors using different training subsets but also through using different distance measures and different random initialization of the regressors parameters. The applicability of the proposed algorithm is broader than standard *Co-Training* algorithm because it does not require multiple redundant and independent views.

2 Co-training by Committee for Regression (*CoBCReg*)

There are two potential problems that can prevent any *Co-Training* style algorithm from exploiting the unlabeled data to improve the performance and these problems are the motivations for this study. Firstly the outputs of unlabeled examples are incorrectly estimated by a regressor that leads to adding noisy examples to the training set of the other regressor. Secondly there is no guarantee that the newly-predicted examples selected by a regressor as *most confident examples* will be *informative examples* for the other regressor. In order to mitigate the former problem, a committee of predictors is used in *CoBCReg* to predict the unlabeled examples instead of a single predictor. For the latter problem, each regressor selects the most informative examples for itself.

Algorithm 1. CoBC for Regression

Require: L - set of m labeled training examples

U - set of n unlabeled examples, T - maximum number of Co-Training iterations

N - number of committee members (*ensemble size*), u - pool size

gr - growth rate, k - number of RBF hidden nodes, α - RBF width parameter

p_i - distance order of the i^{th} regressor

Training Phase

```

1: for  $i = 1$  to  $N$  do
2:    $\{L_i, V_i\} \leftarrow \text{BootstrapSample}(L)$   $\{L_i$  is bag and  $V_i$  is out-of-bag}
3:    $h_i \leftarrow \text{RBFNN}(L_i, k, \alpha, p_i)$ 
4: end for
5: for  $t \in \{1 \dots T\}$  do
6:   if  $U$  is empty then  $T \leftarrow t-1$  and abort loop end if
7:   for  $i \in \{1 \dots N\}$  do
8:     Create a pool  $U'$  of  $u$  examples by random sampling from  $U$ 
9:      $\pi_i \leftarrow \text{SelectRelevantExamples}(i, U', V_i, gr)$ 
10:     $U' \leftarrow U' \setminus \pi_i$  and  $U \leftarrow U \cup U'$ 
11:   end for
12:   for  $i \in \{1 \dots N\}$  do
13:     if  $\pi_i$  is not empty then
14:        $L_i \leftarrow L_i \cup \pi_i$ 
15:        $h_i \leftarrow \text{RBFNN}(L_i, k, \alpha, p_i)$ 
16:     end if
17:   end for
18: end for
```

Prediction Phase

19: **return** $H(x) \leftarrow \sum_{i=1}^N w_i h_i(x)$ for a given sample x

Algorithm 2. SelectRelevantExamples

Require: j - the index of the regressor excluded from the committee

U' - pool of u unlabeled examples V_j - validation set gr - growth rate

```

1: Calculate validation error of  $h_j$  using  $V_j$ ,  $\epsilon_j$ 
2: for each  $x_u \in U'$  do
3:    $H_j(x_u) \leftarrow \frac{1}{N-1} \sum_{i=1, i \neq j}^N h_i(x_u)$ 
4:    $h'_j \leftarrow \text{RBFNN}(L_j \cup \{(x_u, H_j(x_u))\}, k, \alpha, p_j)$ 
5:   Calculate validation error  $\epsilon'_j$  of  $h'_j$  using  $V_j$ , then  $\Delta_{x_u} \leftarrow (\epsilon_j - \epsilon'_j)/\epsilon_j$ 
6: end for
7:  $\pi_j \leftarrow \emptyset$ 
8: for  $gr$  times do
9:   if there exists  $x_u \in U' \setminus \pi_j$  with  $\Delta_{x_u} > 0$  then
10:     $\tilde{x}_j \leftarrow \arg \max_{x_u \in U' \setminus \pi_j} \Delta_{x_u}$ 
11:     $\pi_j \leftarrow \pi_j \cup \{(\tilde{x}_j, H_j(\tilde{x}_j))\}$ 
12:   end if
13: end for
14: return  $\pi_j$ 
```

Let $L = \{(x_i, y_i)\}_{i=1}^m$ and $U = \{x_i\}_{i=1}^n$ represent the labeled and unlabeled training set respectively, which are drawn randomly from the same distribution where y_i is the target real-valued output for each instance x_i in L while the real-valued outputs of instances in U are unknown. The pseudo-code of *CoBCReg* is shown in Algorithm 1. *CoBCReg* works as follow: initially an ensemble consists of N regressors, which is denoted by H , is constructed from L using *Bagging*. Then the following steps will be repeated until the maximum number of iterations T is reached or U becomes empty. For each iteration t and for each ensemble member h_i , a set U' of u examples is drawn randomly from U . The *SelectRelevantExamples* method is applied such that the *companion committee* H_i (ensemble consists of all members except h_i) estimates the output of each unlabeled example in U' . Then h_i is refined using the gr most relevant examples added to its training set L_i . In the prediction phase, the regression estimate for a given instance is the weighted average of the outputs of the regressors created at the final *CoBCReg* iteration.

2.1 Diversity Creation

The combination of an ensemble of regressors is only effective if they are diverse. Clearly, if they are identical, then for each regressor, the outputs estimated by the other regressors will be the same as these estimated by the regressor for itself. That is, there is no more information to be transferred among regressors. Brown et al. presented in [7] an exhaustive survey of the various techniques used for creating diverse ensembles. In regression, ensemble diversity (variance) on an instance x can be quantified by

$$\bar{A}(x) = \sum_{i=1}^N w_i (h_i(x) - H(x))^2. \quad (1)$$

Krogh and Vedelsby [8] introduced the *error-ambiguity decomposition* in which the ensemble error (E) is decomposed into two terms, the weighted average error of the ensemble members (\bar{E}) and the diversity among their outputs for a given instance (\bar{A}). That is, $E = \bar{E} - \bar{A}$. The importance of this decomposition is that it shows us that the average error of the ensemble members should be low while the diversity among them should be high, in order to achieve high ensemble error reduction.

In *CoBCReg*, there are three sources for diversity creation, the *RBF* network regressors are trained using: (1) different bootstrap samples, (2) different random initialization of *RBF* centers and (3) different distance measures. The Minkowski distance between two D -dimensional feature vectors x_1 and x_2 , as defined in (2), is used with different distance order p to train different *RBF* network regressors. In general, the smaller the order, the more robust the resulting distance metric to data variations. Another benefit of this setting, is that, since it is difficult to find in advance the best p value for a given task, then regressors based on different p values might show complementary behavior.

$$\|x_1 - x_2\|_p = \left(\sum_{i=1}^D |x_{1i} - x_{2i}|^p \right)^{1/p} \quad (2)$$

CoBCReg does not hurt the diversity among regressors because the examples selected by a regressor are removed from U . Thus, they can not be selected further by other regressors which keeps the training sets of regressors not similar. Even if the training sets become similar, the regressors could still be diverse because they are instantiated with different distance measures, for some data sets this acts like using different feature spaces.

2.2 Confidence Measure

One of the most important factors that affects the performance of any *Co-Training* style algorithm is how to measure the confidence of a given unlabeled example. The inaccurate confidence estimation can lead to selecting and adding mislabeled examples to the labeled training set and therefore might negatively affect the performance of the *SSL* algorithm. For classification, it is a straightforward task because many classifiers can estimate class posterior probabilities such as Naive Bayes classifier or return real-valued outputs that can be transformed to class probability estimates such as neural networks and decision trees. Assuming that a classifier estimates the probability that an instance x_1 belongs to classes ω_1 and ω_2 is 0.9 and 0.1, respectively, while that for an instance x_2 is 0.6 and 0.4, respectively, then the classifier is more confident that x_1 belongs to classes ω_1 than x_2 . Therefore, a *labeling confidence* can be assigned to each unlabeled example using its class probability distribution.

The main challenge for *CoBCReg* is the mechanism for estimating the confidence because the number of possible predictions in regression is unknown. For regression, in [8], variance is used as an effective selection criterion for active learning because a high variance between the estimates of the ensemble members leads to a high average error. Unfortunately, a low variance does not necessarily imply a low average error. That is, it can not be used as a selection criterion for *SSL* because agreement of committee members does not imply that the estimated output is close to the target output. In fact, we will not measure the *labeling confidence* but we will provide another confidence measure called *selection confidence* (See Algorithm 2). The most relevantly selected example should be the one which minimizes the regressor error on the validation set. Thus, for each regressor h_j , create a pool U' of u unlabeled examples. Then, the root mean squared error (*RMSE*) of h_j is evaluated first (ϵ_j). Then for each example x_u in U' , h_j is refined with $(x_u, H_j(x_u))$ creating new regressor h'_j . So the *RMSE* of h'_j can be evaluated (ϵ'_j), where $H_j(x_u)$ is the real-valued output estimated by the *companion committee* of h_j (H_j denotes all other ensemble members in H except h_j). Finally, the unlabeled example \tilde{x}_j which maximizes the relative improvement of the *RMSE* (Δ_{x_u}) is selected as the most relevant example labeled by *companion committee* H_j .

It is worth mentioning that the *RMSEs* ϵ_j and ϵ'_j should be estimated accurately. If the training data of h_j is used, this will under-estimate the *RMSE*.

Fortunately, since the bootstrap sampling [5] is used to construct the committee, the *out-of-bootstrap* examples are considered for a more accurate estimate of ϵ'_j .

2.3 Two-Phase Learning for RBF Networks

Two-phase learning algorithm of *RBF* network [9] is used for training regressors with *multivariate Gaussian radial basis function* (g) as activation function. At the first phase, the *RBF* centers are determined by performing k -means clustering using the Minkowski distance. The set of Gaussian centers are initialized with training examples randomly selected from L . The width of the j^{th} *RBF* neuron (σ_j) is set to the average Minkowski distance between the center c_j and the two nearest Gaussian centers multiplied by α to control the extent of overlap between them. At the second phase, the output layer weights W which best approximate the limited labeled examples are determined directly by a matrix pseudo-inverse technique, $W = H^+T$, where T is the target outputs of the training examples and H is the activation matrix,

$$H_{ij} = g(x_i; c_j, \sigma_j, p) = g(\|x - c_j\|_p / \sigma_j) = \exp\left(-\frac{\|x - c_j\|_p^2}{2\sigma_j^2}\right) \quad (3)$$

The gradient-descent error backpropagation learning method is not used, otherwise the computational load will be high. On the other hand, direct computation of W is easier and provides instantaneous training of the network. Therefore, the refinement of regressors with newly-labeled examples can be more efficient.

3 Experimental Evaluation

3.1 Methodology

An experimental study is conducted to evaluate *CoBCReg* framework on six data sets described in Table II. *Friedman #1*, *#2*, and *#3* have been used by Breiman [5] for evaluating the performance of *Bagging*. *Gabor* and *Multi* have been used by Hansen [10] for comparing several ensemble methods. *Plane* has

Table 1. Description of the simulated data sets

Data set	Size	Function	Features
<i>Friedman#1</i>	3,000	$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$	$x_1, x_2, x_3, x_4, x_5 \sim U[0, 1]$
<i>Friedman#2</i>	5,000	$y = \sqrt{x_1^2 + (x_2 x_3 - (\frac{1}{x_2 x_4}))^2}$	$x_1 \sim U[0, 100]$ $x_2 \sim U[40\pi, 560\pi]$ $x_3 \sim U[0, 1]$ $x_4 \sim U[1, 11]$
<i>Friedman#3</i>	3,000	$y = \tan^{-1} \frac{x_2 x_3 - (\frac{1}{x_2 x_4})}{x_1}$	$x_1 \sim U[0, 100]$ $x_2 \sim U[40\pi, 560\pi]$ $x_3 \sim U[0, 1]$ $x_4 \sim U[1, 11]$
<i>Gabor</i>	3,000	$y = \frac{\pi}{2} \exp[-2(x_1^2 + x_2^2)] \cos[2\pi(x_1 + x_2)]$	$x_1, x_2 \sim U[0, 1]$
<i>Multi</i>	4,000	$y = 0.79 + 1.27x_1 x_2 + 1.56x_1 x_4 + 3.42x_2 x_5 + 2.06x_3 x_4 x_5$	$x_1, x_2, x_3, x_4, x_5 \sim U[0, 1]$
<i>Plane</i>	1,000	$y = 0.6x_1 + 0.3x_2$	$x_1, x_2 \sim U[0, 1]$

Table 2. Mean and standard deviation of the test $RMSE$ using noise-free functions

Data set	RBFNNs			CoBCReg		
	initial	final	improv	initial	final	improv
Friedman#1	0.0817 ± 0.0042	0.0670 ± 0.0032	17.99%	0.0687 ± 0.0035	0.0590 ± 0.0027	14.12%
Friedman#2	0.0417 ± 0.0033	0.0332 ± 0.0024	20.38%	0.0354 ± 0.0028	0.0294 ± 0.0028	16.95%
Friedman#3	0.1019 ± 0.0037	0.0939 ± 0.0038	7.85%	0.0921 ± 0.0049	0.0863 ± 0.0047	6.08%
Gabor	0.0575 ± 0.0108	0.0330 ± 0.0081	42.60%	0.0375 ± 0.0106	0.0202 ± 0.0062	46.13%
Multi	0.0449 ± 0.0037	0.0345 ± 0.0024	23.16 %	0.0373 ± 0.0038	0.0303 ± 0.0025	18.76%
Plane	0.0180 ± 0.0045	0.0093 ± 0.0032	48.33%	0.0136 ± 0.0045	0.0077 ± 0.0032	43.38%
ave.	0.0576	0.0452	26.72%	0.0474	0.0389	24.24%

been used by Ridgeway et al. [11] for investigating the performance of boosted naive Bayesian regressors. All algorithms are implemented using WEKA library [12]. The input features and the real-valued outputs are scaled to $[0, 1]$. For each experiment, 5 runs of 4-fold cross-validation have been performed. That is, for each data set, 25% are used as test set, while the remaining 75% are used as training examples where 10% of the training examples are randomly selected as the initial labeled data set L while the remaining 90% of the 75% of data are used as unlabeled data set U . In the experiments, an initial ensemble of four *RBF network* regressors, $N = 4$, is constructed by *Bagging* where the distance order p_i used by the i^{th} regressor is set to $i+1$ ($i = 1, 2, 3, 4$). The weights of regressors were uniform, $w_i = 1/N$. we set the pool size u is 50, the growth rate gr is one, the maximum number of iterations T is 30, and for each RBF network the number of *RBFs* k is set to 20 and α is set to 2.0.

3.2 Results

Table 2 present the average of the $RMSE$ s of the four RBF Network regressors used in *CoBCReg* and the $RMSE$ of *CoBCReg* on the test set at iteration 0 (*initial*) trained only on the 10% available labeled data L , after the 30th *SSL* iteration of exploiting the unlabeled data set U (*final*) and the relative improvement percentage on $RMSE$ ($improv = \frac{initial - final}{initial}$). Figure 1 shows the $RMSE$ of *CoBCReg* (*CoBCReg*), and the average of the $RMSE$ s of the four regressors used in *CoBCReg* (*RBFNNs*) at the different *SSL* iterations. The dash and solid horizontal lines show the average of the $RMSE$ s of the four regressors and the $RMSE$ of the ensemble trained using only the 10% labeled data, respectively, as a baseline for the comparison. The dash-dot horizontal line represents the $RMSE$ of the committee trained using all the training data 100% labeled as another baseline. Paired t-test with 0.05 significance level indicates that the final regression estimates of *CoBCReg* are significantly better than its initial estimates on all the data sets. In addition, for all data sets both the *initial* and *final* $RMSE$ of *CoBCReg* (E) (on average, 0.0474 and 0.0389) is less than that of the average of $RMSE$ s of its members (\bar{E}). Therefore, *CoBCReg* can exploit the unlabeled examples to improve the generalization error of the committee and it does not hurt the diversity among the committee members during the *SSL* process ($\bar{A} > 0$).

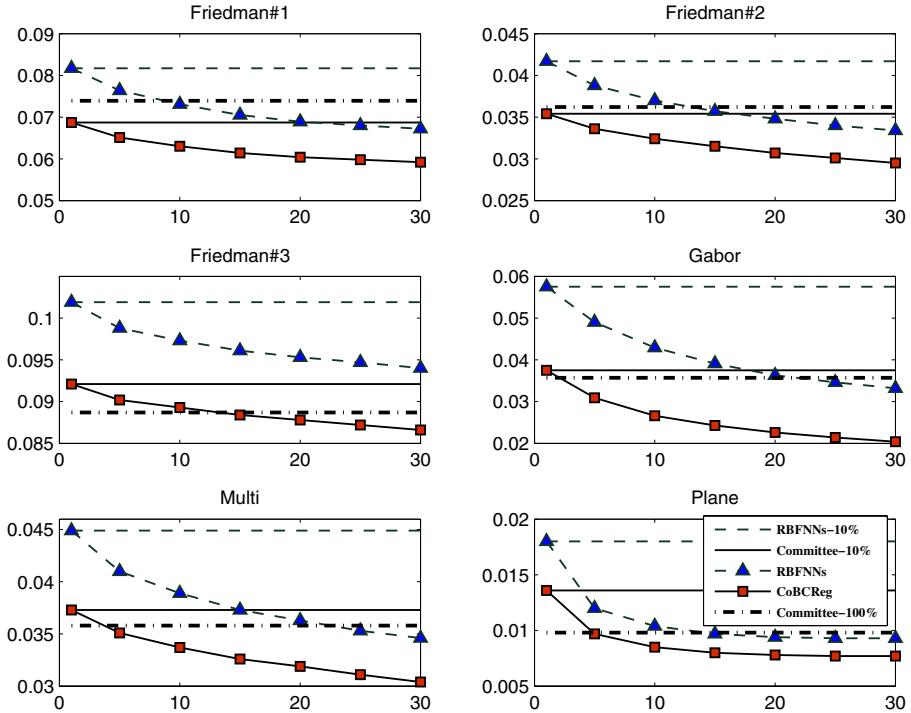


Fig. 1. The average of test *RMSE* at different iterations using noise-free functions

Table 3. Mean and standard deviation of the test RMSE using noisy functions

Data set	RBFNNs			CoBCReg		
	initial	final	improv	initial	final	improv
Friedman#1	0.0860 ± 0.0037	0.0745 ± 0.0026	13.37%	0.0748 ± 0.0035	0.0677 ± 0.0025	9.49%
Friedman#2	0.0669 ± 0.0022	0.0635 ± 0.0013	5.08%	0.0624 ± 0.0016	0.0607 ± 0.0013	2.72%
Friedman#3	0.0962 ± 0.0031	0.0904 ± 0.0029	6.03%	0.0887 ± 0.0036	0.0852 ± 0.0036	3.95%
Gabor	0.0739 ± 0.0073	0.0615 ± 0.0041	16.78%	0.0602 ± 0.0059	0.0541 ± 0.0025	10.13%
Multi	0.0690 ± 0.0029	0.0646 ± 0.0024	6.37%	0.0632 ± 0.0030	0.0607 ± 0.0024	3.96%
Plane	0.0685 ± 0.0055	0.0621 ± 0.0051	9.34%	0.0599 ± 0.0040	0.0592 ± 0.0049	1.34%
ave.	0.0905	0.0770	14.92%	0.0682	0.0646	5.28%

3.3 Influence of Output Noise

In order to study the robustness of *CoBCReg* to noise, we added Gaussian noise to the target functions of *Friedman #1*, *#2*, *#3*, *Gabor*, *Multi* and *Plane* that is distributed as $N(0, 1.0^2)$, $N(0, 125^2)$, $N(0, 0.1^2)$, $N(0, 0.15^2)$, $N(0, 0.35^2)$, and $N(0, 0.05^2)$, respectively, where standard deviation is selected to give 3:1 *signal-to-noise* ratio (i.e., the ratio of the standard deviations). Thus, the variance of the function itself (without noise) accounts for 90% of the total variance. Table 3 present the *initial* and *final* average of the *RMSEs* of the four regressors

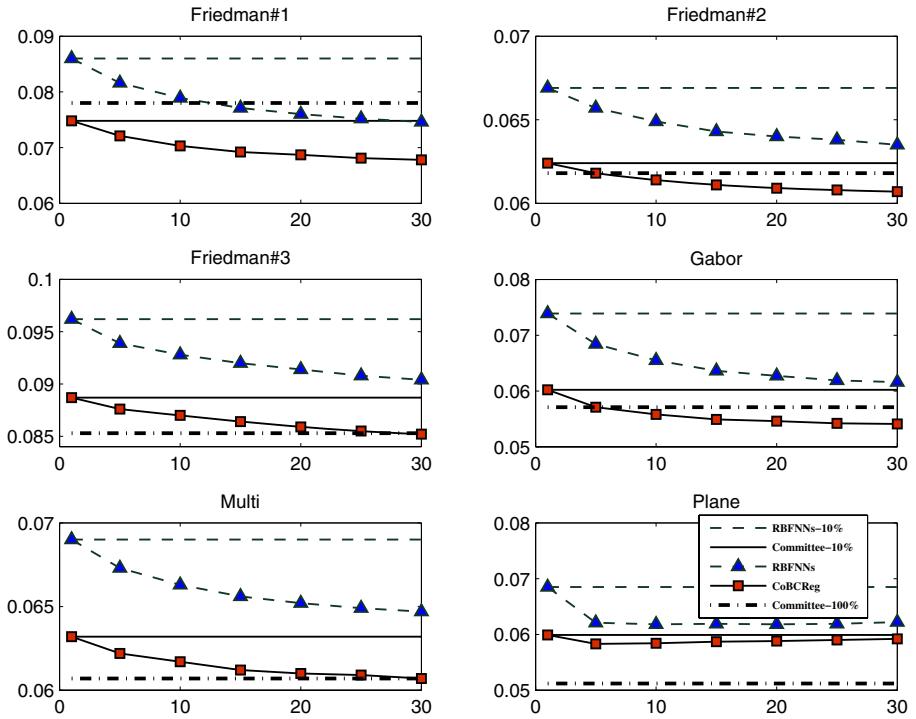


Fig. 2. The average of test $RMSE$ at different iterations using noisy functions

used in *CoBCReg* (\bar{E}) and the $RMSE$ of *CoBCReg* (E) for the noisy functions. Figure 2 shows the performance at the different *SSL* iterations. Again the final regression estimates of *CoBCReg* significantly outperform its initial estimates on all the data sets except on *Plane* where the improvement is not significant. In addition, both the *initial* and *final* E (on average, 0.0682 and 0.0646) is less than that \bar{E} . Although we used *highly noise* problems, *CoBCReg* can still exploit the unlabeled examples to improve the regression estimates on all data sets. It is worth noting that *CoReg*, proposed in [6], was applied on the same data sets and both the absolute $RMSE$ and the relative improvement achieved by *CoBCReg* are better than that of *CoReg* on all data sets.

4 Conclusions and Future Work

For regression tasks, labeling the examples for training is a time consuming, tedious and expensive process. Such burden could be alleviated if the regression learning algorithm can exploit the unlabeled data during learning. In this paper, a *Co-Training* style framework called *CoBCReg* is proposed. It is based on an ensemble of N diverse regressors. At each iteration and for each regressor, the companion committee labels the unlabeled examples then the regressor select

the most informative newly-labeled examples for itself, where the selection confidence is based on estimating the validation error. The final prediction is the average of the estimates of the N regressors. *CoBCReg* is more applicable than the standard *Co-Training* because it does not require sufficient and independent views to construct diverse regressors. However, it depends on three mechanisms to create the diversity, initial regressors are trained using different bootstrap samples with different random initialization of *RBF* centers and are using different Minkowski distance orders. Experimental results show that *CoBCReg* can effectively exploit the unlabeled examples to improve the generalization error.

There are many interesting directions for future work. First, to apply *CoBCReg* using other types of regressors such as linear regressor and support vector regressor. Second, to investigate other diversity creation methods such as using *AdaBoost.RT* [13] ensemble method, an AdaBoost version for regression, that will extend the idea of *CoBCReg*. Third, to explore other confidence measures that are more efficient and effective. Fourth, theoretical analysis of *CoBCReg* is necessary because it will show when and why the algorithm works. Finally, to enhance the performance of *CoBCReg* by interleaving it with *Query by Committee* [8]. Combining semi-supervised learning and active learning within the *Co-Training* setting has been applied effectively for classification.

References

1. Zhu, X.: Semi-supervised learning literature survey. Technical Report (2008)
2. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th Annual Conf. on Computational Learning Theory (COLT 1998), pp. 92–100. Morgan Kaufmann Publishers, San Francisco (1998)
3. Goldman, S., Zhou, Y.: Enhancing supervised learning with unlabeled data. In: Proceedings of the 17th Int. Conf. Machine Learning (ICML 2000), pp. 327–334 (2000)
4. Zhou, Z., Li, M.: Tri-training: Exploiting unlabeled data using three classifiers. IEEE Trans. on Knowl. and Data Eng. 17(11), 1529–1541 (2005)
5. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
6. Zhou, Z.H., Li, M.: Semi-supervised regression with co-training. In: Proceedings of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 908–913 (2005)
7. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. Information Fusion 6(1), 5–20 (2005)
8. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Advances in Neural Information Processing Systems, pp. 231–238. MIT Press, Cambridge (1995)
9. Schwenker, F., Kestler, H., Palm, G.: Three learning phases for radial basis function networks. Neural Networks 14, 439–458 (2001)
10. Hansen, J.: Combining predictors: meta machine learning methods and bias/variance and ambiguity. PhD thesis, University of Aarhus, Denmark (2000)
11. Ridgeway, G., Madigan, D., Richardson, T.: Boosting methodology for regression problems, pp. 152–161 (1999)
12. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (1999)
13. Shrestha, D.L., Solomatine, D.P.: Experiments with adaboost.rt, an improved boosting scheme for regression. Neural Computation 18(7), 1678–1710 (2006)

An Analysis of Meta-learning Techniques for Ranking Clustering Algorithms Applied to Artificial Data

Rodrigo G.F. Soares*, Teresa B. Ludermir, and Francisco A.T. De Carvalho

Federal University of Pernambuco,
Center of Informatics
{rgfs,tbl,fatc}@cin.ufpe.br
<http://www.cin.ufpe.br>

Abstract. Meta-learning techniques can be very useful for supporting non-expert users in the algorithm selection task. In this work, we investigate the use of different components in an unsupervised meta-learning framework. In such scheme, the system aims to predict, for a new learning task, the ranking of the candidate clustering algorithms according to the knowledge previously acquired.

In the context of unsupervised meta-learning techniques, we analyzed two different sets of meta-features, nine different candidate clustering algorithms and two learning methods as meta-learners.

Such analysis showed that the system, using MLP and SVR meta-learners, was able to successfully associate the proposed sets of dataset characteristics to the performance of the new candidate algorithms. In fact, a hypothesis test showed that the correlation between the predicted and ideal rankings were significantly higher than the default ranking method. In this sense, we also could validate the use of the proposed sets of meta-features for describing the artificial learning tasks.

Keywords: Meta-learning, Clustering.

1 Introduction

Selecting suitable algorithms for solving one given problem requires, generally, a great deal of effort. In the context of Machine Learning, we can point out some tasks that one may tackle using more than one technique, such as: classification, regression, clustering. In such domain, there are many alternatives for solving particular problems. This fact raises one of the most difficult tasks in Machine Learning: predicting the performance of candidate algorithms for a given problem. Typically, the choice of which algorithm might be used relies on trial-and-error procedures or on the expensive and rare users' expertise.

Meta-learning approaches have been proposed in order to predict the performance of candidate algorithms for a given problem, so they were able to select

* This work was supported by FACEPE and CNPq - Brazilian funding agencies.

and rank these algorithms indicating to the user the best choices for solving the problem. Such meta-learning techniques offer support to the non-expert user in algorithm selection task, so that there is no need for expertise to deal with this task.

The meta-learning techniques can be used in various domains. One of the main applications of these techniques is in selection and ranking of supervised algorithms. However, just a couple of investigations about the use of these techniques in the unsupervised context were made [20][21]. Those works are the starting points of further researches in that area, although they had a specific case study.

In general, clustering data is a complex task. There are many issues about how clustering can be performed. One single dataset can have more than one cluster structure in different levels of refinement. In fact, there is no even a single definition of what a cluster may looks like. The previous works validated their framework in a particular set of clustering datasets. In this paper, we intend to study and validate the application of meta-learning techniques in the unsupervised context using a wide range of synthetic datasets, covering most of the dataset clustering structures.

In this paper, our aim is to employ the framework proposed in [21][20] with different datasets using a more general set of dataset characteristics, different sets of meta-features in the meta-learning process and a different set of candidate clustering algorithms.

The remainder of this paper is divided into four sections. Section 2 introduces basic concepts about meta-learning and some of its techniques. In Section 3, we present our meta-learning analysis in ranking and selecting clustering algorithms, showing the employed framework, the proposed sets of meta-features and the learning algorithms. Section 4 presents our experiments developed in order to perform the analysis of the unsupervised meta-learning components. Finally, in Section 5, we present some final remarks and further work.

2 Related Works and Basic Concepts

Each meta-example corresponds to a dataset and it is composed of the dataset features (meta-features or meta-attributes) and the information about the performance of one or more algorithms applied to the learning task. The set of meta-examples composes the meta-dataset, which is the input of the meta-learner.

The meta-learner is a system responsible for acquiring knowledge from a set of meta-examples and, then, predicting the performance of the algorithms for new problems. Generally, the meta-features are statistics about the training datasets or some information related to the nature of the data. Examples of these features are: number of training examples, number of attributes, normality tests, number of outliers, among others [13][5][3].

More specifically, each meta-example has, as performance information, a class attribute that indicates the best algorithm for the problem, among a set of candidates [2][14][17][18]. In such a case, the class label for each meta-example is

defined by performing a cross-validation experiment using the available dataset. The meta-learner is simply a classifier which predicts the best algorithm based on the meta-features of the problem.

There is a variety of approaches using meta-learning techniques in literature. For instance, in [12] and [11], different meta-learners are employed to predict a class label associated to the performance of the algorithms, and to recommend a ranking of the algorithms.

Moreover, in the context of unsupervised learning, a novel method was developed to use meta-learning techniques in clustering problems. In [21], the authors presented a novel framework that applies a meta-learning approach to clustering algorithms. Given a dataset, the proposed framework provides a ranking for the candidate algorithms that could be used with that dataset.

Particularly, this paper employs the previous framework in the ranking task of candidate clustering algorithms in a comprehensive range of artificial clustering problems. Additionally, we use two different sets of meta-features in this analysis.

3 Proposed Analysis

In this section, we present the meta-learning process of acquiring knowledge from various datasets and ranking the candidate clustering algorithms.

3.1 General Framework

Figure 1 presents the general architecture of systems used for selecting and ranking clustering algorithms. In order to acquire knowledge and perform the ranking process, the system has two phases: training and use.

In the training phase, the meta-learner (ML) acquires knowledge from the set of examples stored in the database (DB). This knowledge associates dataset features to the performance of the candidate clustering algorithms. The acquired knowledge may be refined as more examples are available in the DB.

In the phase of use, given a new dataset to be clustered, the feature extractor (FE) extracts the values of the dataset features. According to these values, the ML module suggests a ranking of the available candidate algorithms. For that, it uses the knowledge previously provided as a result of the training phase.

The FE module is responsible for extracting the features values of the input datasets. We present these features in the next section.

The DB stores examples of clustering datasets used in the training phase. Each example associates a dataset (represented by the chosen set of features) to the performance of the candidate algorithms. This set of examples is semi-automatically built: (1) the selection of datasets and algorithms to be considered is a manual task; (2) the extraction of the series features is automatically performed by the FE module; and (3) the performance of the candidate algorithms in the clustering of a dataset is empirically obtained by directly applying each algorithm to that dataset and evaluating the resulting clustering structures.

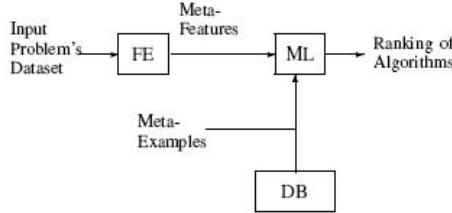


Fig. 1. General architecture

The ML module implements the chosen meta-learning approach to acquiring knowledge (training phase) to be used in the ranking of the candidate algorithms (use phase). The meta-learning approaches implement one or more machine learning algorithms to perform the mentioned task.

3.2 Meta-features

An important issue about implementing the framework is the set of meta-features used by the FE module to describe each dataset. These meta-features depend on the type of dataset under analysis. Some of them are directly related to the nature of the data. In this paper, we use artificial data to evaluate the meta-learning approach, because in this sense we know the clustering data structure. Such fact facilitates the testing and validation of the system (in further work we intend to use real data on the experiments). Then, there is no specific meta-feature describing the data, that is, the set of meta-feature used in this work may be applied in any dataset, since these meta-features are based only on statistics.

Generally, a subjective feature extraction is time consuming, requires expertise, and has a low degree of reliability, such as visual inspection of plots [1]. The presented meta-features are reliably identified, avoiding subjective analysis.

In order to avoid a time consuming selection process, we present a reasonable number of meta-features: nine relevant dataset statistics were used. Some of them were first proposed in [15] for supervised learning.

In this paper, we used two different set of meta-features. Both of them employs the Hotelling's T^2 vector statistics [10]. Given a dataset $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ where $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{im})$, the T^2 vector can be calculated as following:

$$t_i^2 = (\mathbf{x}_i - \bar{\mathbf{x}}) \Sigma^{-1} (\mathbf{x}_i - \bar{\mathbf{x}}), \quad (1)$$

where $\bar{\mathbf{x}}$ is the mean vector of the dataset examples and Σ stands for the covariance matrix. With this equation, one can transform a multivariate dataset into a unidimensional array, condensing the multivariate nature in a single vector [10].

Both sets of meta-features include the following statistics:

1. \log_{10} of the number examples, indicating the amount of available training data.
2. \log_{10} of the number of attributes, indicating the data dimensionality.

3. Multivariate normality test. That is the proportion of T^2 (examples transformed via T^2) that are within 50% of a Chi-squared distribution (degree of freedom equals to the number of attributes describing the example). A rough indicator on the approximation of the data distribution to a normal distribution.
4. Percentage of outliers. This feature is the ratio between the number of t_i^2 farther more than two standard deviations from the mean in the T^2 vector and the total number of dataset examples.

Besides the statistics shown above, the first set of meta-features has five other values.

5. Coefficient of variance (CV) of the first quartile of each attribute. First, for each dataset attribute, we calculate its first quartile. Then, with this vector of quartiles, we compute the standard deviation of such vector and divide it by its mean.
6. CV of the second quartile. It is calculated similarly as the previous feature.
7. CV of the third quartile, computed likewise the previous values.
8. CV of the skewness of the dataset attributes. This value is computed as the previous ones, but considering the skewness of each attribute.
9. CV of the kurtosis of the dataset attributes. This feature takes into account the kurtosis of each attribute, summarizing these measures in a coefficient of variance likewise the previous meta-features.

As shown before, the first set of meta-features, denoted as M_1 , has some meta-features (more precisely, meta-features 5 to 9) that are calculated using univariate statistics: quartiles, skewness and kurtosis. Then, when these measures are calculated for each attribute of a dataset, a vector of the analyzed statistic is generated. Since we do not use modal meta-features to compose the meta-dataset, we must use a single value to describe one given characteristic of a learning task dataset. Thus, in this work, in order to summarize the multivariate nature of the datasets, such meta-features rely on the simple coefficient of variance.

We also analyzed a second set of meta-features, denoted as M_2 . This set have the same first four meta-features as presented before. The last five values are based on the T^2 vector. They are calculated as the three quartiles, skewness and kurtosis of the T^2 vector. Instead of using the coefficient of variance to summarize these statistics, it is expected that such vector is able to retrieve more properly the multivariate information of the data.

3.3 Algorithms

In the framework analyzed here, we must define the candidate clustering algorithms that will be applied on each learning task. An appropriate set of clustering techniques might have algorithms with different types of internal mechanisms, so that this set can deliver a variety of performances in the clustering task.

The selected candidate algorithms are: single linkage (SL), complete linkage (CL), average linkage (AL), k-means (KM), Shared Nearest Neighbors (SNN),

mixture model clustering (M), farthest first (FF), DB-scan (DB) and x-means algorithm (XM) [9,23,6,7,16].

The ranking of algorithms is a more informative way of selecting algorithm [21]. Then, using the meta-learner, we intend to predict the rank of each clustering algorithm according to the quality of the partitions generated by the algorithm under analysis. Each meta-example is labeled with nine values according to the rank of each candidate algorithm. Given a dataset to be labeled, a label is set to 1 if the corresponding algorithm had the best performance, 2 if it had the second best performance and so on, until the value 9 is given to the worst algorithm. The system uses a average ranking to deal with ties, for instance, if the two best algorithms are tied, their rank is set to 1.5.

The global error rate was used as the performance criterion since it allows a fair comparison between the clustering algorithms runs [22]. Such measure is simply the proportion of examples that fall outside the cluster that corresponds to its actual class. We consider that two algorithm are tied if the difference between them, in terms of global error rate, is less than 0.01.

Once we have composed the meta-dataset with the meta-features and the rankings, we now can define the meta-learner: the learning system that will associate the dataset characteristics (meta-features) to the ranking of the algorithms for predicting the rankings for new datasets. We analyzed two different learning methods as the meta-learner.

The first meta-learner is the Multilayer Perceptron network (MLP) used as a regressor of the rankings. The predictions are taken directly from the output nodes of the network. Each output node is responsible for delivering the rank of the corresponding algorithm, yielding the predicted ranking vector.

The Support Vector Regression (SVR) [19] were also employed as meta-learner. In this case, one SVR is trained for predicting the rank of each clustering algorithm, thus the system have nine independent regressors. The outcome of the SVR-based meta-learner is a vector with the predictions of each candidate algorithm.

4 Experiments

4.1 Description of the Datasets

In order to evaluate the proposed methodology, we generated 160 artificial datasets with the data generators available in [8]. We aim to obtain datasets in a wide representative range of cluster structures, so that the system can properly learn the performance of the algorithms.

The first generator is based on a standard cluster model using multivariate normal distributions. The Table I shows the parameter setup of the gaussian cluster generator. For each of the 8 combinations of cluster number and dimension, 10 different instances were generated, giving 80 data sets in all.

Due to the lack of generality of spherical clusters, we employed a second alternative cluster generator that delivers more elongated cluster shapes in arbitrarily

Table 1. Parameter setup for the gaussian cluster generator

Parameter	Range
Number of clusters	2,4,8,16
Dimension	2,20
Size of each cluster	uniformly in [10, 100] for 2 and 4 cluster instances, and [5, 50] for 8 and 16 cluster instances.

high dimensions. This second generator creates ellipsoidal clusters with the major axis at an arbitrary orientation.

The ellipsoid cluster generator delivers sets of high dimension. The Table 2 presents the parameters of the ellipsoid cluster generator. For each of the 8 combinations of cluster number and dimension, 10 different instances were generated, giving 80 data sets in all.

Table 2. Parameter setup for the ellipsoid cluster generator

Parameter	Range
Number of clusters	2,4,8,16
Dimension	50,100
Size of each cluster	uniformly in [10, 100] for 2 and 4 cluster instances, and [5, 50] for 8 and 16 cluster instances.

4.2 Evaluating the System

We executed 30 runs of each non-deterministic candidate algorithms. The number k of clusters was set to the actual class number of each dataset. We evaluate the performance of the meta-learners using the leave-one-out procedure.

The quality of a suggested ranking for a given dataset is evaluated by measuring the similarity to the ideal ranking, which represents the correct ordering of the models according to the global error rate. We employed the Spearman's rank correlation coefficient [3] to measure the similarity between a suggested and the ideal rankings.

In order to calculate this coefficient, we compute, given a meta-example i , the sum of squared differences between the predicted and ideal rankings for each clustering algorithm j as shown in the Equation 2

$$D_i^2 = \sum_j D_{ij}^2 \quad (2)$$

And then, the average of Spearman's coefficient for the 160 meta-examples is calculated using the Equation 3

$$SRC = \frac{1}{160} * \sum_{i=1}^{160} \left\{ 1 - \frac{6 * D_i^2}{P^3 - P} \right\} \quad (3)$$

where P is the number of candidate algorithms. The value of this coefficient ranges from $[-1, 1]$. The larger is the value of SRC_i , the greater is the similarity between the suggested and the ideal rankings for the dataset i .

In our implementation, we used the WEKA software to execute the MLP regressors [22] and the regression Support Vector Machine (SVR) algorithm, implemented in LIBSVM: a library for support vector machines [4].

4.3 Results

As highlighted before, we used two sets of meta-features, giving two different meta-datasets to analyze. These datasets were applied to both MLP and SVR regressors used as meta-learners. The results of both meta-learners were compared to the default ranking method. In such method, the average rank of each algorithm is suggested for every test example.

For the first meta-dataset, M_1 , the Table 3 shows the means and standard deviations of SRC for each meta-learner.

Table 3. Results of the meta-learners for the M_1 meta-dataset

Meta-learner	Mean	Standard deviation
MLP	0.886	0.138
SVR	0.850	0.153
Default ranking	0.846	0.142

And for the second meta-dataset, formed by statistics of the T^2 vector, the Table 4 presents the results obtained by the tested meta-learners.

Table 4. Results of the meta-learners for the M_2 meta-dataset

Meta-learner	Mean	Standard deviation
MLP	0.891	0.137
SVR	0.883	0.152
Default ranking	0.846	0.142

For both datasets, M_1 and M_2 , the rankings predicted by the MLP and SVR methods were more correlated to the ideal rankings than the default ranking method. A hypothesis test at a significance level of 5% showed that the mean of the correlation values of both MLP and SVR meta-learners were statistically higher than that obtained with the default ranking. However, there was no relevant difference between the correlation values of MLP and SVR methods for both meta-datasets. Both meta-features sets were validated using the tested meta-learners. However, further investigations about the choice of such features can lead to an improvement of the overall performance of the system.

5 Final Remarks

In this work, we employed different regression methods as meta-learners in a meta-learning approach. We also proposed two sets of meta-features. One has meta-features based on statistics extracted directly from the data and the other has meta-features calculated from the T^2 vector.

In order to evaluate the meta-learning techniques in a more comprehensive way, we used synthetic data with a wide range of cluster structures. Moreover, we applied nine well-known clustering algorithms with different internal mechanisms.

We were able to validate the use of both sets of meta-features in describing the unsupervised artificial datasets, allowing the meta-learners to successfully associate these characteristics to the performance of the clustering algorithms.

Since we could successfully instantiate the meta-learning framework described before and validate its use in artificial data, an even more comprehensive work can be done by applying other clustering algorithms, or other learning methods as meta-learners. Additionally, one can apply this meta-learning approach to datasets from other contexts.

An important issue that can have further analysis is the set of meta-features. Such issue is a open investigation point, causing a great impact in the final result of the system and it is not trivially obtained: these measure cannot rely on the class attribute.

References

1. Adya, J.A.M., Collopy, F., Kennedy, M.: Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting* 17(2), 143–157 (2001)
2. Aha, D.W.: Generalizing from case studies: A case study. In: *Proceedings of the Ninth International Workshop on Machine Learning*, pp. 1–10. Morgan Kaufmann, San Francisco (1992)
3. Brazdil, P.B., Soares, C., Da Costa, J.P.: Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning* 50(3), 251–277 (2003)
4. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
5. Engels, R., Theusinger, C.: Using a data metric for preprocessing advice for data mining applications. In: *European Conference on Artificial Intelligence*, pp. 430–434 (1998)
6. Ertoz, L., Steinbach, M., Kumar, V.: A new shared nearest neighbor clustering algorithm and its applications. In: *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*, pp. 105–115 (2002)
7. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) *Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231. AAAI Press, Menlo Park (1996)

8. Handl, J., Knowles, J.: Cluster generators for large high-dimensional data sets with large numbers of clusters (2008), <http://dbkgroup.org/hndl/generators>
9. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
10. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis, 5th edn. Prentice Hall, Englewood Cliffs (2002)
11. Kalousis, A., Hilario, M.: Feature selection for meta-learning. In: Cheung, D., Williams, G.J., Li, Q. (eds.) PAKDD 2001. LNCS (LNAI), vol. 2035, pp. 222–233. Springer, Heidelberg (2001)
12. Kalousis, A., Theoraris, T.: Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. Intelligent Data Analysis 3(5), 319–337 (1999)
13. Kalousis, A., Gama, J., Hilario, M.: On data and algorithms: Understanding inductive performance. Machine Learning 54(3), 275–312 (2004)
14. Kalousis, A., Hilario, M.: Representational issues in meta-learning. In: ICML, pp. 313–320 (2003)
15. Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J.: Machine learning, neural and statistical classification. Ellis Horwood, Upper Saddle River (1994)
16. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Seventeenth International Conference on Machine Learning, pp. 727–734. Morgan Kaufmann, San Francisco (2000)
17. Prudêncio, R.B.C., Ludermir, T.B., de A.T. de Carvalho, F.: A modal symbolic classifier for selecting time series models. Pattern Recognition Letters 25(8), 911–921 (2004)
18. Prudêncio, R.B.C., Ludermir, T.B.: Meta-learning approaches to selecting time series models. Neurocomputing 61, 121–137 (2004)
19. Chen, P.H., Fan, R.E., Lin, C.J.: Working set selection using the second order information for training svm. Journal of Machine Learning Research 6, 1889–1918 (2005)
20. Soares, R.G.F.: The use of meta-learning techniques for selecting and ranking clustering algorithms applied to gene expression data (in portuguese). Master's thesis, Federal University of Pernambuco - Center of Informatics (2008)
21. Souto, M.C.P., Prudêncio, R.B., Soares, R.G.F., Araújo, D.A.S., Filho, I.G.C., Ludermir, T.B., Schliep, A.: Ranking and selecting clustering algorithms using a meta-learning approach. In: IEEE (ed.) Proceedings of International Joint Conference on Neural Networks, pp. 3729–3735 (2008)
22. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
23. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks 16(3), 645–678 (2005)

Probability-Based Distance Function for Distance-Based Classifiers

Cezary Dendek¹ and Jacek Mańdziuk²

¹ Warsaw University of Technology, Faculty of Mathematics and Information Science,
Plac Politechniki 1, 00-661 Warsaw, Poland
dendekc@studentmini.pw.edu.pl

² Warsaw University of Technology, Faculty of Mathematics and Information Science,
Plac Politechniki 1, 00-661 Warsaw, Poland
Tel.: (48 22) 621 93 12; Fax: (48 22) 625 74 60
mandziuk@mini.pw.edu.pl
<http://www.mini.pw.edu.pl/~mandziuk/>

Abstract. In the paper a new measure of distance between events/observations in the pattern space is proposed and experimentally evaluated with the use of *k*-NN classifier in the context of binary classification problems. The application of the proposed approach visibly improves the results compared to the case of training without postulated enhancements in terms of speed and accuracy.

Numerical results are very promising and outperform the reference literature results of *k*-NN classifiers built with other distance measures.

1 Introduction

The problem of constructing and measuring a distance between observations is frequently encountered in numerous application fields. The usual approach is based on constructing distance functions in the directly observed space (usually \mathbb{R}^n).

However, in many cases of real-life problems the dimensions of observations are mappings of probability space (eg. biological data often expresses genome and history of an individual). Exploration of the probability space features and their inclusion in the distance measure, often based on correlation between dimensions (*Mahalanobis distance*) and their individual influence on classification accuracy (*weighted distances*), usually increases accuracy. Effectively, those improvements change the space of observations or the space of their difference using linear and quadratical transformations.

In this paper another direction is explored: measurement of distance in partially reconstructed and standardized probability space. The model of distance is proposed in section 2. The benchmark data sets and results of numerical evaluation of proposed distance measure efficacy in the context of *k*-NN classifier are presented in sections 3 and 4, respectively. Conclusions and directions for future research are placed in the last section.

Presented measure of distance is a direct continuation and generalization of authors' previous work [1] introducing probability-related distance measure and works [2][3] related to properties of metrical structure of pattern space.

2 Distance in the Training Patterns Space

2.1 Introduction

Pattern space has naturally defined structure of metrical space which is obtained by its immersion into \mathbb{R}^n . This approach however does not preserve structure of probability space, which can be used to improve accuracy of estimators.

Improved immersion can be obtained with the use of *Cumulative Density Functions* (CDF) by transformation of pattern space, as described in [1]. Let CDF_i denotes CDF calculated on i -th dimension of pattern space. Transformation of pattern is defined as follows:

$$(CDF(x))_i := CDF_i(x_i)$$

Application of CDF transformation on pattern space creates standardized space (denoted *CDF-Space*). Projection of training patterns into *CDF-Space* results in uniform distribution of patterns in each dimension (marginal distributions are $U[0, 1]$).

Estimation of *CDF* (denoted as *ECDF*) can be obtained either by parametric estimation (fitting parameters of arbitrary chosen family of distributions) or by the use of simple non-parametric estimator as following:

$$ECDF_i(x) = \frac{|\{z_i \in TrSet : z_i \leq x_i\}|}{|TrSet|},$$

where $TrSet$ denotes the training set.

2.2 Model of Distance

Structure of the introduced distance measure consists of two components: *univariate distance measure* (discussed in Sect 2.5) providing a measure of distance in a given dimension and *linking function* (discussed in Sect 2.6) which combines those measures and provides a univariate distance. Proposed distance measures are applicable to probabilistic spaces of structure S_e , introduced in Sect 2.3.

2.3 Event Manifestation Error

Let D be probability distribution of univariate continuous random variable V over probability space S . The idea of proposed *probabilistic distance* is based on modification of standard sampling process: event E in space S does not map directly to value v of function V ($v = V(E)$) – instead, the *error of event manifestation*, denoted e , is introduced: E maps to the neighborhood of v according to the distribution of the error e and distribution D :

$$v = V(E \oplus e)$$

Presented approach can be opposed to one usually used:

$$v = V(E) \oplus e$$

Proposed model creates new probability space S_e .

The process of sampling the space S can be expressed in terms of CDF of distribution D by simple conversion of S by mapping its events to $U[0, 1]$ distribution and using inverse theorem. Random variable V can be sampled in the following way:

$$V = CDF^{-1}(U[0, 1])$$

Let *error of event manifestation* be a random variable of distribution Err . The process of sampling the space S_e can be expressed as:

$$V = CDF^{-1}(\min(\max(U[0, 1] + Err, 0), 1))$$

2.4 Model of Probabilistic Distance

Let v be an observation of a random variable V and x be a fixed point, $v, x \in \mathbb{R}$. As a distribution Err of *event manifestation error* $U[-1, 1]$ has been chosen for the sake of simplicity.

The *probabilistic distance from fixed point x to observation v* is a probability measure of smallest neighborhood of x , generated by the manifestation error e and containing v . In terms of CDF it can be expressed as:

$$d(x; v) = \int_{CDF^{-1}(x_c - |x_c - v_c|)}^{CDF^{-1}(x_c + |x_c - v_c|)} dCDF(x) = \min(1, x_c + |x_c - v_c|) - \max(0, x_c - |x_c - v_c|),$$

where $x_c = CDF(x)$ and $v_c = CDF(v)$. As the postulated measure is a probability, $d(x; v) \geq 0$.

The contour plot of function $d(x; v)$ is presented in Fig 2.4. The value of $d(x; v)$ can be regarded as the proportion of events for which the distance from x is not greater than the distance between x and v . As the consequence, the value of an expression

$$p_{Eq}(x; v) = 1 - d(x; v)$$

can be regarded as the p-value of a hypothesis that $x = v$ vs $x \neq v$.

2.5 Univariate Distance Measures in CDF-Space

A distance measure on the CDF-Space is required to be symmetrical and to operate on observations rather than fixed values. This goal can be achieved by combination of $d(x; v)$ values. The contour plots of all proposed variants of the distance are presented in Fig 2.5. The following variants has been considered:

Distance Based on Expected Value. Let ExpVal be defined as follows:

$$\text{ExpVal}(u, v) = CDF^{-1}\left(\frac{CDF(u) + CDF(v)}{2}\right)$$

The value of the function is the conditional expected value of the random variable X distributed according to CDF :

$$E(X | X \in [u, v]) = \text{ExpVal}(u, v)$$

The distance variant expressing the assumption that both observed events, u and v are manifestations of theirs average value can be introduced as follows:

$$D_{\text{ExpVal}}(u, v) = \frac{d(\text{ExpVal}(u, v); v) + d(\text{ExpVal}(u, v); u)}{2}$$

Expression could be simplified as:

$$D_{\text{ExpVal}}(u, v) \propto d(\text{ExpVal}(u, v); v) + d(\text{ExpVal}(u, v); u) \propto |CDF(u) - CDF(v)|$$

Function ExpVal provides as immersion of the classical distance in the model of the probabilistic one. The obtained simplified form has been evaluated in details in [1].

Distance Based on min Function

$$D_{\text{Min}}(u, v) = \min(d(u; v), d(v; u))$$

This variant expresses the assumption that more distant observation is the manifestation of less distant one.

Distance Based on max Function

$$D_{\text{Max}}(u, v) = \max(d(u; v), d(v; u))$$

This variant expresses the assumption that less distant observation is the manifestation of more distant one. It has been introduced in order to provide the contrast for the function D_{Min} .

Distance Based on the Distance Average

$$D_{\text{Avg}}(u, v) = \frac{d(u; v) + d(v; u)}{2} \propto d(u; v) + d(v; u) = D_{\text{Max}}(u, v) + D_{\text{Min}}(u, v)$$

Distance Based on the Independency Assumption

$$\begin{aligned} D_{\text{Ind}}(u, v) &= 1 - \left(1 - d(u; v)\right) \left(1 - d(v; u)\right) = 1 - p_{\text{Eq}}(u, v)p_{\text{Eq}}(v, u) = \\ &= d(v; u) + d(u; v) - d(v; u)d(u; v) \end{aligned}$$

This variant joins the p-values of two nonsymmetric tests of equality into one, using the assumption that tests are independent.

Distance Based on the Carthesian Sub-Linking

$$D_{\text{Cart}}(u, v) = \sqrt{d(u; v)^2 + d(v; u)^2}$$

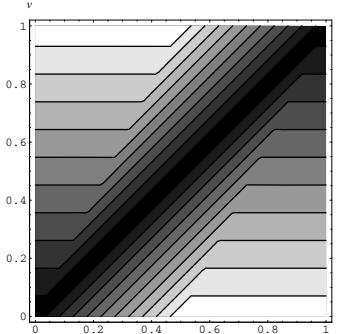


Fig. 1. Probabilistic distance from fixed point x to observation v

2.6 Linking Function

In order to provide a unified distance measure for pattern space in case of multidimensional data, the distances calculated independently in each dimension have to be combined. The combination, defined as *linking function*, can be parametrically dependent on the training set (and, usually, more computationally requiring) or data-independent. Let $D_i(x_i, y_i)$ denotes a distance measures in i -th dimension and $C(x, y)$ the combined distance measure.

Standard Linking. This data-independent variation is based on Carthesian distance definition:

$$C_{\text{std}}(x, y) = \sum_{i=1}^n D_i(x_i, y_i)^2$$

Mahalanobis Linking. This data-dependent variation is based on Mahalanobis distance definition, which includes an information of estimated observations' covariances between sub-dimensions. Let Σ denotes covariance matrix of $CDF_i(x_i)$. A distance between events x and y is defined as:

$$C_{\text{Mah}}(x, y) := \sqrt{\left([D_i(x_i, y_i)]_{i=1}^n \right)^T \Sigma^{-1} \left([D_i(x_i, y_i)]_{i=1}^n \right)}$$

Average Linking

$$C_{\text{avg}}(x, y) = \frac{1}{n} \sum_{i=1}^n D_i(x_i, y_i)$$

Mahalanobis-Avg-SQRT Linking. Let Σ denotes covariance matrix of $CDF_i(x_i)$. A distance between events x and y is defined as:

$$C_{\text{MahAvgSqrt}}(x, y) := \frac{1}{n} \sum_{i=1}^n \left(\Sigma^{-\frac{1}{2}} [D_i(x_i, y_i)]_{i=1}^n \right),$$

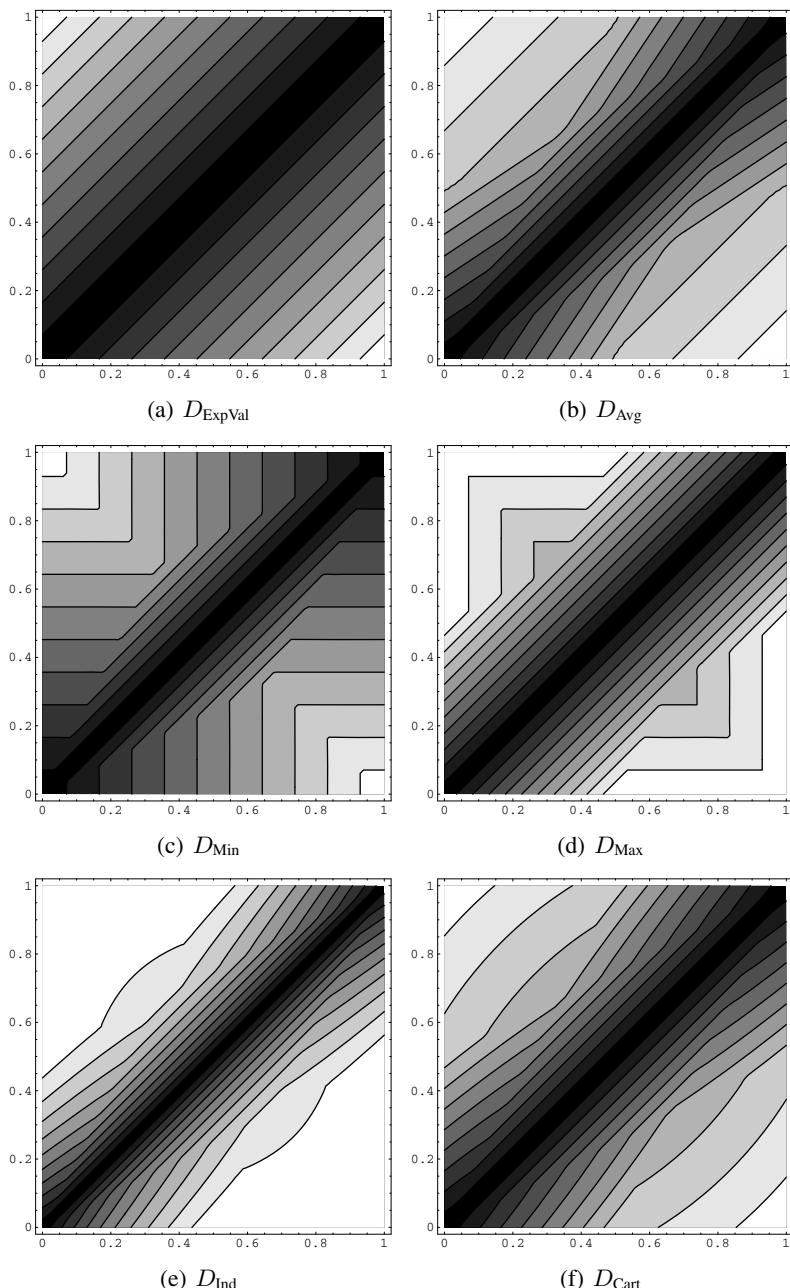


Fig. 2. Contour plots of distances between $(u, v) \in [0, 1]^2$ calculated with postulated probabilistic distances

3 Data Sets

In order to provide experimental support of presented method validity and generate results that can be compared to other sources, data sets available at *UCI Machine Learning Repository* [4] were chosen. The benchmark sets were selected according to the following criteria and reasons:

- they represent a binary classification problem
- the number of observations is lower than 1 000 (due to the high computational cost of leave-one-out method of estimating the accuracy)
- they represent integer, real or binary categorical type of attributes in order to avoid discrete distance problem

A brief characteristics of selected data sets is presented in Table I

Table 1. A brief characteristics of selected data sets

data set	instance number	attributes number	class proportion
<i>BUPA Liver Disorders</i>	345	7	200 : 145
<i>Pima Indians Diabetes</i>	768	9	500 : 268
<i>Wisconsin Diagnostics Breast Cancer</i>	569	31	357 : 212
<i>Sonar</i>	208	61	111 : 97
<i>Ionosphere</i>	351	34	225 : 126

The data sets used in the following experiments are obtained by the transformation described in sect. 2.1 with the use of the simple, nonparametric estimation of *ECDF* which resulted in normalization of data and uniformity of marginal distributions. In order to provide the possibility to assess the significance of experimental results, the raw, untransformed data sets has been used where indicated.

4 Results

All data sets has been evaluated with the use of a k -NN classifier for each combination of the univariate distance measure and the link function defined in sections 2.5 and 2.6 resp. Results of misclassification rate estimation (in per cent) have been obtained with use of the *leave-one-out* estimator.

In order to extend the results described in [1] and to provide the context of the current research data sets were evaluated *with* and *without* application of outlier removal procedure proposed in [1]. Results of the performed evaluation provide the possibility of assessing the sensitivity of introduced distance measures to the presence of outliers.

Numerical results of the evaluation of the proposed distance components are presented in Tables 4 and 2 respectively. Presented results are misclassification rates of the best (in terms of the highest accuracy of the k -NN classifier) distance function constructed with use of a given distance component and evaluated in the context of a given

data set. The *overall ranking* column contains the rank of the given component in the ranking of the average ranks of the components. The average ranks are calculated by averaging ranking of the respective components in the context of a particular data set.

Table 3 provides the comparison with k -NN classifier based on standard, Euclidean distance.

The results presented in the Tables fully confirm the efficacy of the proposed distance construction: for each evaluated data set the best obtained model, concerning a misclassification rate estimation, is the model constructed using the model and components proposed (Tables 2 and 3). High observed rank of D_{ExpVal} component, which can be reduced to a simple difference of CDF , empirically shows the significance of other univariate distance measures (Tables 2 and 4). The sums of rank results of all univariate distance components except for D_{ExpVal} and D_{Max} are similar. The observed noticeable variance of the individual rank results for each component provides the direction for future work.

Finally, the advantage of the application of the outlier removal procedure has been clearly shown.

In summary, the preliminary works within the model of probability-based distance measure (with and without the use of outliers removal algorithm) allowed, within the k -NN models, to achieve comparable or better results than the best ones presented in

Table 2. k -NN 1–CV minimal misclassification rate estimation for the distance built with the given distance element

distance element	data set										overall ranking	
	Pima		Bupa		WDBC		Sonar		Ionosphere			
	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN		
D_{Avg}	26.95	25.78	32.46	30.43	2.81	3.34	12.98	11.54	9.12	9.69	4	
D_{Cart}	27.21	25.39	32.46	30.14	2.81	3.51	12.5	12.02	8.83	9.69	3	
D_{Min}	27.47	25.52	31.3	29.57	2.64	2.99	11.54	12.98	9.4	9.69	1	
D_{Ind}	25.52	26.04	32.46	29.57	2.99	2.99	13.46	13.46	8.26	8.26	2	
D_{ExpVal}	27.34	26.43	33.33	29.57	2.28	2.81	12.98	11.06	9.69	<i>11.11</i>	5	
D_{Max}	26.69	25.13	33.33	<i>31.01</i>	3.16	3.69	14.42	15.38	9.12	10.26	6	
C_{Avg}	26.69	25.13	32.46	30.14	2.28	2.81	12.98	11.06	8.26	8.26	1	
$C_{\text{MahAvgSqrt}}$	27.99	26.43	32.46	29.57	2.28	2.81	12.98	11.06	8.55	8.26	2	
C_{std}	25.52	26.04	33.62	29.57	2.99	2.99	11.54	11.54	9.12	9.12	3	
C_{Mah}	27.47	27.34	31.3	29.57	2.99	2.99	11.54	11.54	9.12	8.83	3	

Table 3. k -NN 1–CV misclassification rate estimation for the Carthesian distance

space	data set											
	Pima		Bupa		WDBC		Sonar		Ionosphere			
	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN		
\mathbb{R}^n –space	30.60	28.52	36.23	33.62	7.38	6.68	18.27	17.31	15.10	15.38		
ECDF–space	27.73	26.56	35.07	31.59	2.99	3.69	12.98	13.94	19.09	22.22		

Table 4. k -NN 1–CV minimal misclassification rate estimation for the distance built with the given distance element after the outlier extraction

distance element	data set										overall ranking	
	Pima		Bupa		WDBC		Sonar		Ionosphere			
	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN	3–NN	5–NN		
D_{Avg}	21.88	22.27	27.83	28.12	2.64	3.51	12.98	11.54	8.55	9.12	1	
D_{Cart}	22.27	22.01	28.41	28.12	2.64	3.69	12.5	11.54	8.26	8.83	1	
D_{Min}	22.92	22.92	26.67	27.54	2.64	3.16	11.54	12.02	8.83	9.12	3	
D_{Ind}	22.53	22.92	27.25	28.12	2.99	3.16	13.46	12.5	7.98	7.98	4	
D_{ExpVal}	23.31	22.92	29.28	28.41	2.28	2.99	12.98	11.06	9.69	<i>10.83</i>	5	
D_{Max}	22.66	22.92	29.28	28.41	2.99	3.51	14.42	15.38	8.83	9.97	6	
C_{Avg}	21.88	22.27	29.28	28.7	2.28	2.99	12.98	11.06	7.98	7.98	1	
$C_{\text{MahAvgSqrt}}$	23.05	22.92	29.86	28.12	2.28	2.99	12.98	11.06	7.98	7.98	2	
C_{std}	22.53	22.01	28.99	28.12	<i>3.16</i>	3.16	11.54	12.02	8.83	8.55	3	
C_{Mah}	23.44	22.92	26.67	27.54	<i>3.16</i>	3.16	11.54	12.02	8.55	8.55	4	

Table 5. k -NN misclassification rate comparison. The results in the first two rows are calculated with probability-based distance measure, *with* and *without* outlier removal, respectively. Results in the following rows are taken from the literature. *Non. avail.* denotes that the results could not be found it the respective papers.

distance measure	estimation method	BUPA	Pima	WDBC	Sonar	Ionosp.
probability-based <i>with</i> outlier removal (Table 4)	leave-one-out CV	26.67	21.88	2.28	11.06	7.98
probability-based <i>without</i> outlier removal (Table 2)	leave-one-out CV	29.57	25.13	2.28	11.06	8.26
adaptive distance measure [5]	leave-one-out CV	30.59	25.13	2.79	12.00	4.29
cam weighted distance [6]	leave-one-out CV	35.3	24.7	3.5	<i>Non. avail.</i>	6.8
weighted distances [7]	100 x 5–CV	36.22	27.33	<i>Non. avail.</i>	<i>Non. avail.</i>	<i>Non. avail.</i>
boosting distance estimation [8]	100 x 20/80	33.58	28.91	4.67	25.67	16.27

the literature. In particular the following comparisons with other definitions of the metric functions within the considered data sets has been taken into account: weighted distances [7], adaptive distance measure [5], boosting distance estimation [8] and cam weighted distance [6]. Table 5 presents the summary of results accomplished by the k -NN classifiers.

5 Conclusions

In the paper a new class of measures of distance between events/observations in the pattern space is proposed and experimentally evaluated with the use of k -NN classifier in the context of binary classification problems. It is shown that proposed measures

produce in average better results than training without their use in all of the evaluated cases. Crossvalidational estimate of resulting model quality has been compared with numerical results provided by other researchers, concerning the k -NN classifiers built with other distance measures. Other possible applications of presented distance measure (especially in the context of training sequence construction proposed in [2]) as well as separate selection of univariate distance for each dimension are considered as future research plans.

Acknowledgement

This work was supported by the research grant from the Warsaw University of Technology.

References

1. Dendek, C., Mańdziuk, J.: Improving performance of a binary classifier by training set selection. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 128–135. Springer, Heidelberg (2008)
2. Dendek, C., Mańdziuk, J.: Including metric space topology in neural networks training by ordering patterns. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4132, pp. 644–653. Springer, Heidelberg (2006)
3. Mańdziuk, J., Shastri, L.: Incremental class learning approach and its application to handwritten digit recognition. Inf. Sci. Inf. Comput. Sci. 141(3-4), 193–217 (2002)
4. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
5. Wang, J., Nesovic, P., Cooper, L.N.: Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recogn. Lett. 28(2), 207–213 (2007)
6. Zhou, C.Y., Chen, Y.Q.: Improving nearest neighbor classification with cam weighted distance. Pattern Recogn. 39(4), 635–645 (2006)
7. Pardes, R., Vidal, E.: Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(7), 1100–1110 (2006)
8. Amores, J., Sebe, N., Radeva, P.: Boosting the distance estimation. Pattern Recogn. Lett. 27(3), 201–209 (2006)

Constrained Learning Vector Quantization or Relaxed k-Separability

Marek Grochowski and Włodzisław Duch

Department of Informatics, Nicolaus Copernicus University, Toruń, Poland
grochu@is.umk.pl

Abstract. Neural networks and other sophisticated machine learning algorithms frequently miss simple solutions that can be discovered by a more constrained learning methods. Transition from a single neuron solving linearly separable problems, to multithreshold neuron solving k -separable problems, to neurons implementing prototypes solving q -separable problems, is investigated. Using Learning Vector Quantization (LVQ) approach this transition is presented as going from two prototypes defining a single hyperplane, to many co-linear prototypes defining parallel hyperplanes, to unconstrained prototypes defining Voronoi tessellation. For most datasets relaxing the co-linearity condition improves accuracy increasing complexity of the model, but for data with inherent logical structure LVQ algorithms with constraints significantly outperforms original LVQ and many other algorithms.

1 Introduction

Problems with Complex logical structure are difficult to solve with feedforward neural networks, such as the Multilayer Perceptrons (MLPs), the basis set expansion (Radial Basis Function, RBF) networks, or Support Vector Machines (SVMs). The k -separability index [1] breaks the class of non-separable classification problems into subclasses that require at least k intervals for separation of data after a single linear projection (equivalent to k parallel hyperplanes). Such problems are called k -separable, with $k = 2$ for linear separability. High values of k characterize problems that are difficult to learn for feedforward neural networks and kernel classifiers [2], although in principle they may be solved by a single multi-threshold neuron. For example, the d -bit parity problem is very difficult because it is $d + 1$ -separable [3], with vectors forming compact clusters only after suitable projection, not before. Many problems in bioinformatics, text analysis and other fields have inherent complex logic that makes learning difficult. Changing the cost function from a typical error function evaluating separability [2] to function that rewards discovery of interesting patterns in the data helps to create good internal representations. For k -separable problems reward for creation of large clusters of pure vectors after projection to the hidden space creates features that greatly simplify the learning process [3].

Direct use of linear projections is not always the best idea. For example, image segmentation requires distinguishing complex shapes defined in two- or three-dimensional space. This type of difficulty is solved by using localized kernels to project original

data to the high-dimensional space [4], providing flexible decision borders in the original space. However, for highly non-separable problems with non-compact distribution of data clusters, as is the case of Boolean functions, flexible decision borders are not sufficient, and significant simplifications may be achieved if instead of linearly separable k -separable learning target is defined [1]. Such approach may discover simple and comprehensible data models, but needs to be extended if simple solutions do not exist.

In this paper systematic increase of model complexity is investigated, from the simplest, single neuron perceptron that solves separable problems, through k -separable projections equivalent to constrained vector quantization methods, to unconstrained prototype-based LVQ solutions. This is achieved by investigating relations between projection-based and distance-based solutions. In the first case constructive neural networks with perceptron-type nodes may be used [5][6], while in the second case localized function networks, or learning vector quantization methods, provide prototype-based rules [7]. In the next section some drawbacks of existing systems are briefly pointed out and relations between k -separable solutions and constrained Learning Vector Quantization (cLVQ) approaches are explored. Section three describes two such cLVQ algorithms that are tested on some benchmark datasets in section four.

2 k -Separability and Prototypes

There is a growing evidence that existing algorithms easily miss simple solutions for various datasets. Consider the Australian Credit dataset [8] that has been used in the Statlog project [9]. 690 cases of credit card applications are characterized using 6 numerical and 8 categorical attributes, with some missing values. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. Almost the same number of approval (55.5%) as disapproval cases (44.5%) has been selected. In the Statlog project [9] over 20 classifiers have been applied to this data, with the best results reached in 10-fold crossvalidation by the C4.5 decision trees, around $87 \pm 4\%$ accuracy with an average of about 6 nodes per tree. Other trees showed an average of about 85.5% and lower, with as many as 70 nodes. Radial Basis Function (RBF) networks gave also about 85.5%, and MLP about 1% lower, while linear SVM gives $85.2 \pm 3.0\%$, with over 190 support vectors. Optimized Gaussian-kernel SVM is slightly better, with $86.2 \pm 4.1\%$, for optimized $C = 0.1, \sigma = 0.01$, leaving over 460 support vectors [1]. LVQ gives here even worse results at the 80% accuracy level. Unfortunately all these learning methods missed the simplest rule that has $\text{Acc}=85.5 \pm 4.0\%$ accuracy, based on a single binary feature: if A8=True then Yes, if A8=False then No. This solution is statistically not worse than the best results for this dataset, and much more comprehensible than those provided by alternative methods. Regularization in neural networks [2] does not provide bias towards this type of simple solutions, preferring many small weights instead of binarized weights.

All learning methods mentioned above also fail when sharp decision borders are needed. On most data for which logical rules gave good results [10] neural networks, LVQ, SVM and the nearest neighbor methods are much worse than decision trees. For example, the Thyroid Disease data set [8], with 3772 examples for learning (primary

¹ Calculations performed with the Ghostminer package, www.fqs.pl/ghostminer

hypothyroid, compensated hypothyroid, normal, 2.47%, 5.06%, 92.47% respectively), and 3428 for testing (similar class distribution) has 21 attributes (15 binary, 6 continuous). Rules based on 4 continuous and 2 logical features give 99.36% accuracy on the test set, while the best neural approaches are not better than 98.5% and SVM with optimized Gaussian kernel achieves only 96.1%. Results on complex Boolean problems are even worse, but in this case also decision trees fail completely [56].

Thus in a number of relatively simple problems popular machine learning methods are not able to find good models. One may of course develop a methodology that would start from simple rules [10], for example highly pruned decision trees, and proceed to more complex learning algorithms testing different methods. However, it would be better to have a model that could deal with such problems directly. To this aim relations between projection and distance-based methods are explored below.

A linearly separable data is divided by a hyperplane into two distinct classes. The \mathbf{w} vector is perpendicular to this hyperplane and defines a direction for the projection $z(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} / \|\mathbf{w}\|$ of d -dimensional data points \mathbf{x} , mapping all points from one class on $z < \theta$, and from the other class on $z > \theta$. The conditional probability densities $p(z|C)p(C)$ after projection of the training data may be used to draw a histogram and estimate probability distributions (Fig. 1). Identical solution is obtained using two prototypes $\mathbf{t}_i, i = 1, 2$ placed on the \mathbf{w} line symmetrically around the threshold $\theta = |z(\mathbf{t}_1) - z(\mathbf{t}_2)|/2 = |z_1 - z_2|/2$, each associated with its class C_i , and a rule: choose C_1 if $|z(\mathbf{x}) - z_1| < |z(\mathbf{x}) - z_2|$, else choose C_2 . However, the interpretation of such prototypes may be problematic, as their placement is not unique and cannot in general match both probability density peaks. To facilitate arbitrary placements each prototype should have its own threshold θ_i , with $z_1 + \theta_1 = \theta = z_2 - \theta_2$. Then the decision rule becomes: C_1 if $z(\mathbf{x}) < z_1 + \theta_1$, or C_2 if $z(\mathbf{x}) < z_2 - \theta_2$. Using two instead of one threshold allows also to define a confidence margin around θ where probability densities may overlap decreasing reliability of predictions.

Some non-separable problems are k -separable; a projection exists that gives k pure clusters of vectors that belong to different classes. It requires only $d + k - 1$ parameters, one direction \mathbf{w} and $k - 1$ intervals. Most Boolean problems for d -bit strings have $k \sim n/2$ with maximum k probably equal to $d + 1$ [1]. This is equivalent to k collinear LVQ prototypes $\mathbf{t}_i, i = 1..k$. For example, in d -bit parity problem projection on the direction of the main diagonal $\mathbf{w} = [1, 1, \dots, 1]$ creates clusters for alternating odd and even-parity classes containing strings with 0, 1, 2 ... d bits equal to 1, separated from each other by $\sqrt{n}/(n + 1)$. The decision rule says: if $z(\mathbf{x}) \in [\theta_{i-1,i}, \theta_{i,i+1}]$ than \mathbf{x} belongs to the class associated with z_i prototype, where $\theta_{i-1,i}$ is the Bayesian threshold calculated from probability distributions for $z(\mathbf{x})$ for vectors that fall into adjacent clusters. Prototypes $z_i = z(\mathbf{t}_i; \mathbf{w})$ may fall at any point in this interval.

Even though a single projection may sometimes be sufficient for correct classification a larger number of good projections may be useful for building a robust predictor. This may be achieved by sequential constructive network algorithms using projection pursuit based on the Quality of Projected Clusters (QPC) [56]. Note that solutions provided by MLP, RBF or SVM classifiers for k -separable problems are much more complex and cannot generalize well. If not all data samples are separated by projections extraction of information based on similarity should be tried. One of the simplest such

transformations is done by finding prototypes in the original data space, for example by the Learning Vector Quantization (LVQ) algorithm [11][12], and then use distances to these prototypes as new dimensions. The number of dimensions is then equal to the number of prototypes. LVQ algorithm may also be used to look for optimal projections, but there are two problems with this solution. First, k unconstrained prototypes require kd parameters, while for k -separable problems models $d + k - 1$ parameters are sufficient. Second, it is very difficult to find good unconstrained solution for problems that are highly-nonseparable. Even in the Australian Credit problem LVQ has failed to find binary feature that separates the data fairly well. Finding optimal scaling factors for features used in the distance function is quite hard, therefore all distance-based methods, including LVQ, easily miss it.

In the spirit of the structural risk minimization approach [13] data models of progressively increasing complexity will be created by adding prototypes and relaxing constraints. The simplest model searches for separable solutions starting from testing projections on each of the input features x_i . A line $w_{ij} = m_i - vm_j$ connecting means m_i of vectors for each class for Gaussian distributions gives optimal solution for pairwise class discrimination. This requires d parameters for w plus one parameter for the threshold. In general optimal direction may be found using the QPC criterion [5]. Fixing the reference point at m_1 mean similar direction $w \sim m_1 - t_1$ should be found by the LVQ optimization of a single prototype t_1 . k -separable solutions are only slightly more complex, defining along $z(x)$ projection line k intervals and requiring $d + k - 1$ parameters. This is equivalent to k co-linear prototypes t_i , with one prototype per resulting cluster. LVQ algorithm should find similar solution if a prototype t_1 is adapted without constraints, providing direction $w \sim m_1 - t_1$. The remaining $k - 1$ prototypes are placed on this line, adapting only their positions $z_i = w \cdot t_i$.

Decision borders created by co-linear prototypes divide whole feature space with parallel hyperplanes. If this solution is not sufficient constraints should be relaxed to allow for LVQ adaptation of all prototypes. This creates Voronoi tessellation using kd parameters. Allowing for adaptation of the scaling factors and the type of distance functions creates even more complex decision regions [14][15], but we shall not investigate it here. Starting from one prototype, adding more prototypes with constraints, and finally allowing for full LVQ adaptation, should systematically explore different biases leading to progressively more complex models. Constraints of this type are a form of regularization [2] that should help to find solutions that are not easy to discover.

3 Constrained Learning Vector Quantization (cLVQ)

The constructive cLVQ algorithm combines (non-local) projection pursuit scheme and (local) LVQ learning. Learning starts from the simplest classification model (in terms of k -separability), defined by a single projection and a single prototype per class. For two-class linearly separable problems this is sufficient. If this is not the case the number of parameters of the model is increased until results will show no significant improvement. Complexity of the model may be increased in two ways: by adding consecutive projections, or by increasing the number of prototypes.

The cLVQ model may take the form of a two layer feedforward neural network. Let $\mathcal{W} = \{w_i\}, i = 1, \dots, h$ denote weights of connections between the d input and

the h hidden nodes, performing linear projections $\mathcal{W}\mathbf{x}$, scaling and combining input features. Let $\mathcal{T} = \{\mathbf{t}_j\}, j = 1, \dots, k$ denote connection between hidden to the k output nodes with each weight t_{jl} coding positions of prototypes \mathbf{t}_j after projection \mathbf{w}_l . Classification is done using the winner takes all (WTA) rule, where for a given input \mathbf{x} the network returns class label associated with prototype \mathbf{t}_r most similar to \mathbf{x} : $r = \arg \min_j \|\mathcal{W}\mathbf{x} - \mathbf{t}_j\|$. Fully connected network has $(d + k)h$ parameters, where d is the number of inputs, k the number of prototypes (output nodes) and h is the number of projections (hidden nodes). Removing connections with small weight values may further reduce model complexity.

This model is trained by two approaches: first, maximization of the QPC projection pursuit index to find projection directions (QPC-LVQ), and second, based on the first PCA eigenvector (PCA-LVQ). The cLVQ network learning starts with an empty hidden layer and k (the number of classes) output nodes. First optimization of projections \mathbf{w}_i that separate data clusters is described. Centers of these clusters are determined by positions of prototypes \mathbf{t}_j placed along the projected line $\mathbf{w}_i^T \mathbf{x}$. Positions of these prototypes are adjusted using LVQ1 algorithm [11], with additional term that attracts prototypes to the line. Position of the winning prototype \mathbf{t} (closest to the training vector \mathbf{x}) is adjusted using the following formula:

$$\Delta \mathbf{t} = \alpha \delta(C_{\mathbf{x}}, C_t)(\mathbf{x} - \mathbf{t}) + \beta ((\mathbf{t} - \boldsymbol{\mu}) - \mathbf{w}^T(\mathbf{t} - \boldsymbol{\mu})\mathbf{w}) \quad (1)$$

The first term of Eq. 1 scaled by coefficient α , comes from standard LVQ1 procedure, where $C_{\mathbf{x}}$ denotes class of the vector \mathbf{x} and $\delta(C_{\mathbf{x}}, C_t) = +1$ if $C_{\mathbf{x}} = C_t$ and -1 otherwise. The second term, scaled by β , pushes prototypes in the direction that brings them closer to the $\mathbf{x} = \mathbf{w}z + \boldsymbol{\mu}$ line, where \mathbf{w} defines a direction in space and $\boldsymbol{\mu}$ is a certain point on the line, for example a mean position of all prototypes $\boldsymbol{\mu} = \frac{1}{N} \sum \mathbf{t}_i$. For $\beta = 0$ Eq. 1 reduces to the original LVQ1 algorithm. If $\beta = 1$ the second term of Eq. 1 will put the prototypes exactly along the line, i.e. this corresponds to the orthogonal projection of prototypes on the direction \mathbf{w} . The effective number of parameters is reduced by adding these constraints.

The first PCA eigenvector calculated on co-linear prototypes (PCA-LVQ) is equivalent to fitting points to a line defining projection direction \mathbf{w} . Another approach is based on the Quality of Projected Clusters (QPC) index, a supervised algorithm that finds interesting linear projections for multiclass datasets [6]. Assuming that a set of prototypes has been created by the LVQ procedure the QPC index is defined as:

$$QPC(\mathbf{w}) = \sum_{i=1}^n \left(A_i \sum_{j=1}^k \delta(C_{\mathbf{x}_i}, C_{\mathbf{t}_j}) G(\mathbf{w}^T(\mathbf{x}_i - \mathbf{t}_j)) \right) \quad (2)$$

where $G(x)$ is a function with localized support and maximum in $x = 0$ (e.g. a Gaussian function), $C_{\mathbf{x}_i}$ is the class of vector \mathbf{x}_i , $C_{\mathbf{t}_j}$ the class of prototype \mathbf{t}_j , and $\delta(C_{\mathbf{x}_i}, C_{\mathbf{t}_j}) = \pm 1$. This index achieves maximum value when linear projection on the direction defined by \mathbf{w} groups vectors \mathbf{x}_i from a single $C_{\mathbf{x}_i}$ class into compact clusters close to the prototype \mathbf{t}_j associated with this class. Parameters A_i control influence of each term in Eq. 2. If $A_i = A^+$ for all i that satisfies $C_{\mathbf{x}_i} = C_{\mathbf{t}_j}$ and $A_i = A^-$ otherwise, then large value of A^- enforces strong separation between clusters, while A^+

has influence mostly on compactness and purity of clusters. Optimization of QPC index is performed by a simple gradient-based method. Since it may require a large number of iterations to converge, to speed up learning one can search for w only after some specified number of learning epochs during LVQ optimization phase.

Procedure 1 describes steps performed to expand the network, creating a sequence of hidden nodes, where weights w_i of each successive node represent unique direction in the input space according to the projection pursuit scheme.

Procedure 1. (Construction of the sequence of hidden nodes)

1. start with k output nodes and no hidden nodes;
2. add a new hidden node, optimize its weights w_i and positions of prototypes t_j (see Procedure 2);
3. if the error does not decrease return the network which gives the lowest error;
4. else increase the number of prototypes splitting the node that makes the largest number of errors;
5. orthogonalize training data to the i -th projection, and repeat, starting from step 2.

Each new projection w_i is trained separately on dataset that is orthogonal to all directions w_1, \dots, w_{i-1} found earlier. Another way to gain unique projections is to perform learning only on a subset of training samples misclassified by the network. Only weights of one hidden node w_i are adjusted in step 2, all projections found earlier are fixed. Each change of weights w_i requires proper modification of weights between the i -th hidden node and all outputs. Number of output nodes needs to be optimized as well, because each new direction w_i produces different input space projection image. The region covered by the prototype t_j that leads to the largest number of errors (defined by the WTA rule) is divided by adding new prototype associated with the class that has most errors (step 3).

Procedure 2 describes optimization of hidden nodes w_i .

Procedure 2. (Searching for an optimal hidden node after adding a new hidden node.)

1. optimize weights w_i and prototype positions t_j to find the best projection that minimizes classification error (using PCA-LVQ or QPC-LVQ algorithms);
2. remove redundant prototypes using condensed nearest neighbor (CNN) method;
3. adjust positions of the remaining prototypes (LVQ learning in the hidden space);
4. if the current projection does not lead to lower error than projections already found return;
5. else increase the number of prototypes splitting the output node that makes the largest number of errors, and go to step 2;

The final network after learning may contain prototypes which are useless (e.g. LVQ learning can expel them far from the training data points). Redundant prototypes that do not decrease accuracy are removed using Hart's condensed nearest neighbor (CNN) method [16][17], where prototypes are used as reference vectors. Positions of the remaining prototypes are then adjusted with few steps of LVQ to improve cluster separation in the hidden space (after $w \cdot x$ projection).

The cost of computations of Eq. II is $O(knd)$ for QPC-LVQ network and $O(knd^2)$ for PCA-LVQ, where d denotes dimensionality of the data, n is the number of instances

and k the number of prototypes. For each LVQ learning step the winner prototype must be chosen, which requires $O(nkd)$ operations. Only one hidden node is adapted at a time and the computation of QPC index is linear respect to number of instances, number of prototypes and dimensionality of data. Both LVQ and QPC may require trying a few initializations and performing many iterations, thus evaluation of QPC (or PCA, respectively) should be done only after some LVQ learning steps. Removal of redundant prototypes using CNN method also requires $O(knd)$ time. Thus the overall computational cost of searching for optimal solution by the methods described above should not exceed $O(knd)$ for QPC and $O(knd^2)$ for PCA, respectively.

4 Results

Results of constrained LVQ models – QPC-LVQ and PCA-LVQ – are compared to the original LVQ, support vector machine (SVM), k-nearest-neighbors (kNN) and the multilayer perceptron network (MLP) classifiers. Accuracy of classification has been estimated using 10 fold stratified cross-validation method for several datasets with different types of features. Following datasets from the UCI repository [8] have been used: Appendicitis, Australian, Breast Cancer Wisconsin, Glass, Heart, Ionosphere, LED (with 500 generated samples), Ljubljana Breast Cancer, Voting and Wine, plus the melanoma skin cancer data collected in the Outpatient Center of Dermatology in Rzeszów in Poland [18]. In addition two Boolean artificial datasets have been used: 10-dimensional parity problem, and a Mirror Symmetry dataset with 1000 strings of 20 bits, where half of them are symmetric. This parity problem is 11-separable while the symmetry problem is 3-separable, but requires a set of exponentially growing weights. Results are collected in Table 4. For QPC-LVQ and PCA-LVQ average number of projections (#P) and number of prototypes (#K) is reported, for LVQ number of prototypes (#K, the size of codebook), for SVM average number of support vectors (#SV), and for MLP networks average number of hidden neurons (#N) are reported (where 0 means no hidden neurons).

SVM with Gaussian kernel was used with γ and C parameters fully optimized using an inner 5-fold crossvalidation procedure (the Ghostminer software was used²), MLP networks were trained with error backpropagation several times changing the number of hidden nodes from 0 to 20, and selecting the network that gave the best training results. For k -class problems MLP networks with k outputs were used (1-of- k output coding), except for two classes when a single output was used. For LVQ the number of prototypes (codebook vectors) was increased from 1 to 20, and only the best solution is reported. The optimal number of neighbors for kNN method was estimated each time during 5-fold inner crossvalidation, and the best one model was chosen for training. Euclidean measure was used to calculate distance to neighbors.

The SVM is quite accurate for almost all UCI datasets, but failed completely for the parity problem. In most cases large number of support vectors are created even when simple solutions exists. MLP was able to handle all problems with good accuracy and a small number of neurons, but at a rather high cost, requiring many initializations with increasing number of hidden nodes from 0 to 20. Setting the right number of hidden

² <http://www.fqs.pl/ghostminer/>

Table 1. Average classification accuracy for 10-fold crossvalidation test (see text for explanation); bold numbers = accuracy within variance from the best result

Dataset	QPC-LVQ			PCA-LVQ			LVQ			SVM			kNN			MLP		
	acc.	#P	#K	acc.	#P	#K	acc.	#K	acc.	#SV	acc.	#K	acc.	#K	acc.	#N		
Appendicitis	86.1 \pm 8.5	1.0	2.5	82.2 \pm 7.4	1.0	3.1	86.6 \pm 6.8	2	86.7 \pm 10.8	31.4	84.0 \pm 6.1	4.8	87.8 \pm 10.1	3				
Australian	86.1 \pm 4.1	1.0	2.0	86.1 \pm 4.1	1.0	2.0	85.6 \pm 4.6	2	84.9 \pm 1.6	206.1	85.1 \pm 2.9	8.2	86.8 \pm 4.9	2				
Mirror Symmetry	87.3 \pm 4.2	1.2	3.5	78.1 \pm 11.4	1.8	5.3	75.7 \pm 4.3	12	98.0 \pm 1.4	296.5	89.3 \pm 2.6	5.8	92.1 \pm 10.7	3				
Breast C.W.	96.2 \pm 1.8	1.0	2.0	96.2 \pm 2.1	1.0	2.0	96.5 \pm 2.5	2	96.6 \pm 1.7	51.1	97.1 \pm 1.4	5.6	96.9 \pm 1.0	1				
Melanoma	85.7 \pm 6.2	1.0	4.0	70.8 \pm 9.0	1.7	4.3	76.8 \pm 7.2	4	85.2 \pm 5.7	240.3	86.0 \pm 7.4	1.0	94.4 \pm 2.7	3				
Glass	60.4 \pm 10.3	1.1	4.5	58.9 \pm 7.3	1.4	4.0	66.3 \pm 10.5	7	64.9 \pm 6.2	283.6	68.8 \pm 9.2	1.4	69.2 \pm 10.4	7				
Heart	78.9 \pm 8.5	1.0	2.0	80.7 \pm 9.0	1.0	2.0	82.2 \pm 8.6	2	81.5 \pm 9.1	101.5	78.5 \pm 7.6	8.5	79.6 \pm 9.8	1				
Ionosphere	78.4 \pm 8.0	1.0	3.1	75.5 \pm 8.5	1.1	3.5	81.9 \pm 7.0	4	93.5 \pm 4.7	61.0	84.0 \pm 7.7	1.2	81.5 \pm 5.5	5				
Iris	96.0 \pm 4.4	1.0	3.0	94.7 \pm 4.0	1.0	3.0	97.3 \pm 3.3	3	96.7 \pm 4.7	39.6	94.5 \pm 6.9	5.8	95.3 \pm 5.2	2				
L. Breast	72.6 \pm 4.4	1.0	2.0	74.0 \pm 6.9	1.0	2.1	74.7 \pm 6.0	3	73.3 \pm 9.6	143.6	73.7 \pm 5.5	6.9	72.2 \pm 6.7	1				
LED500	58.5 \pm 8.0	1.1	9.6	45.5 \pm 8.6	1.5	8.2	72.0 \pm 5.1	10	65.2 \pm 5.5	664.1	71.2 \pm 6.6	8.5	65.2 \pm 7.0	0				
Parity 10	96.1 \pm 3.9	1.0	6.8	97.6 \pm 0.8	1.0	7.5	51.3 \pm 4.9	8	44.2 \pm 5.7	921.2	80.7 \pm 3.4	20.0	82.9 \pm 17.2	14				
Voting	95.2 \pm 3.5	1.0	2.0	90.6 \pm 5.5	1.3	2.5	93.8 \pm 2.7	5	95.9 \pm 2.4	57.0	93.3 \pm 3.2	4.6	95.9 \pm 2.4	1				
Wine	96.0 \pm 5.7	1.9	3.1	94.9 \pm 4.7	2.0	3.0	97.7 \pm 2.8	4	96.6 \pm 2.9	63.7	95.0 \pm 4.1	6.2	96.6 \pm 2.7	2				

nodes has been very important to reach high quality solutions but also reduces time to convergence. The number of parameters adapted during training of MLPs with one hidden layer is $(d + 1)h + (h + 1)k$, where d denotes number of inputs, h - number of hidden nodes and k the number of output nodes.

The QPC-LVQ algorithm was able to find in most cases, with very few hidden nodes, solution of comparable accuracy to all other classifiers. Moreover, for some datasets (mirror symmetry, melanoma and parity) it has significantly outperformed original LVQ1. Only for the LED data the QPC-LVQ model has been too simple and gave worse results. The PCA-LVQ performs worse in most cases, comparison of PCA-LVQ and QPC-LVQ using Wilcoxon [19] test shows significant differences, at confidence level of 95%, in terms of accuracy (p -value of 0.048) and number of parameters (p -value of 0.003) both in favor of QPC-LVQ. QPC-LVQ compared with LVQ, SVM and kNN shows no significant difference in accuracy, giving p -value equal to 0.62, 0.07 and 0.87, respectively. Only for the MLP Wilcoxon test find significant differences (p -value of 0.044) against QPC-LVQ. On the other hand MLP and LVQ needed on average 3 times more parameters than QPC-LVQ.

5 Discussion

Drawbacks of existing learning methods that fail on relatively simple problems, as well as on highly-non separable problems, may be remedied by structural risk minimization approach, in which models of growing complexity are introduced by relaxing various constraints. One such approach has been analyzed here, based on the constrained LVQ model, allowing for systematic relaxation of constraints to create progressively more complex solutions. Such models may learn optimal solutions to complex Boolean functions as well as problems that are almost linearly separable using single feature, while unconstrained LVQ converges to worse solutions. Two versions of such approaches have been presented here, based on PCA and QPC algorithm to find good projection lines. Straightforward LVQ modification allows for reduction of the number of effective

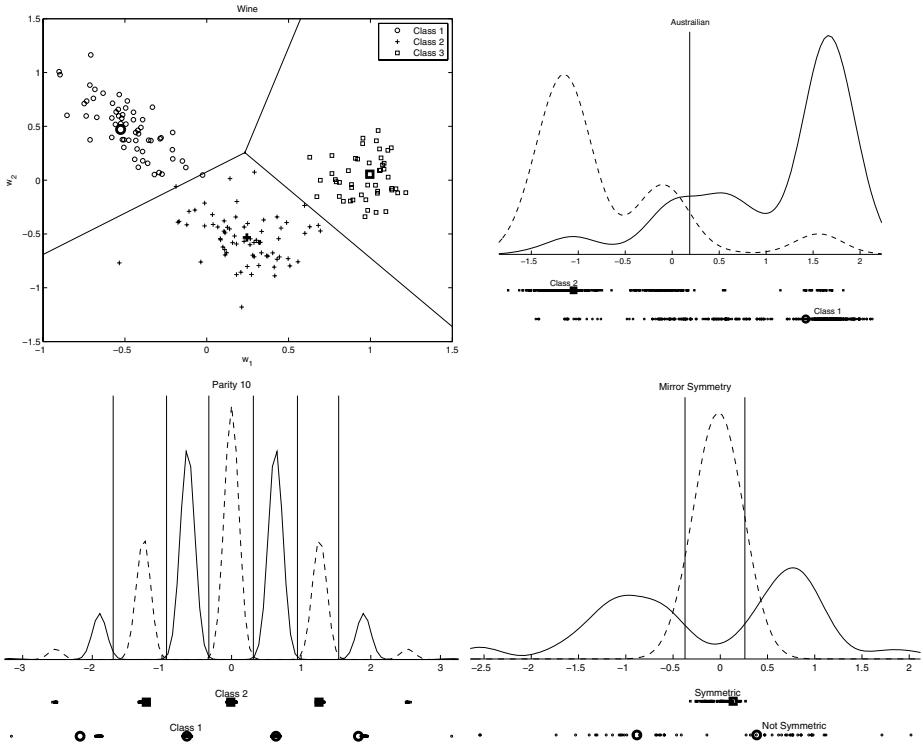


Fig. 1. Visualization of hidden layer of the QPC-LVQ network for Wine, Australian, Parity 10 and the Mirror Symmetry dataset

parameters by forcing the LVQ prototypes to stay near those projection lines. In tests one or two lines have usually been sufficient, with very small number of prototypes. Great advantage of algorithms developed here is that the hidden layer contains only one or two nodes, and therefore activations of those nodes may be used to visualize inherent dataset relations. Fig. 1 illustrates this, with prototypes and decision boundaries set by prototypes (the winner-takes-all rule leads to Voronoi tessellation) for Wine, Australian, Parity 10 and the Mirror Symmetry datasets. Such visualizations allows to estimate confidence of classification for each vector. The main point of this paper was to show that the strategy of starting with constrained models and gradually relaxing constraints (in Eq. 1 decreasing β to 0) should help to discover simple models that more complex approaches will miss, and also in some complex cases find better solutions. This is indeed observed in Tab. 4: for some data LVQ-QPC has indeed advantages, while for other data that require more complex decision borders unconstrained optimization is significantly more accurate. Thus it is recommended to always use methods that allow for such control of model complexity.

References

1. Duch, W.: *k*-separability. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 188–197. Springer, Heidelberg (2006)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
3. Grochowski, M., Duch, W.: Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 180–189. Springer, Heidelberg (2007)
4. Schölkopf, B., Smola, A.: Learning with Kernels. In: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
5. Grochowski, M., Duch, W.: A Comparison of Methods for Learning of Highly Non-Separable Problems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 566–577. Springer, Heidelberg (2008)
6. Grochowski, M., Duch, W.: Projection Pursuit Constructive Neural Networks Based on Quality of Projected Clusters. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 754–762. Springer, Heidelberg (2008)
7. Duch, W., Blachnik, M.: Fuzzy rule-based systems derived from similarity to prototypes. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) ICONIP 2004. LNCS, vol. 3316, pp. 912–917. Springer, Heidelberg (2004)
8. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
9. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine learning, neural and statistical classification. Elis Horwood, London (1994)
10. Duch, W., Setiono, R., Zurada, J.: Computational intelligence methods for understanding of data. Proceedings of the IEEE 92(5), 771–805 (2004)
11. Kohonen, T.: Self-organizing maps. Springer, Heidelberg (1995)
12. Biehl, M., Ghosh, A., Hammer, B.: Dynamics and generalization ability of lvq algorithms. J. Mach. Learn. Res. 8, 323–360 (2007)
13. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
14. Duch, W., Adamczak, R., Diercksen, G.: Classification, association and pattern completion using neural similarity based methods. Applied Mathematics and Computer Science 10, 101–120 (2000)
15. Duch, W.: Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics 29, 937–968 (2000)
16. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 114, 515–516 (1968)
17. Jankowski, N., Grochowski, M.: Comparison of instance selection algorithms. i. algorithms survey. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 598–603. Springer, Heidelberg (2004)
18. Hippe, Z.: Data mining in medical diagnosis. In: Kącki, E. Computers in Medicine, Łódź, Poland, vol. 1, pp. 25–34. Polish Society of Medical Informatics (1999)
19. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics 1, 80–83 (1945)

Minimization of Quadratic Binary Functional with Additive Connection Matrix

Leonid Litinskii

Center of Optical Neural Technologies Scientific Research Institute for System Analysis,
Russian Academy of Sciences,
44/2 Vavilov Str, Moscow 119333, Russia
litin@mail.ru

Abstract. $(N \times N)$ -matrix is called additive when its elements are pair-wise sums of N real numbers a_i . For a quadratic binary functional with an additive connection matrix we succeeded in finding the global minimum expressing it through external parameters of the problem. Computer simulations show that energy surface of a quadratic binary functional with an additive matrix is complicate enough.

1 Introduction

In the present paper we analyze the classic problem of discrete mathematics that is minimization of a quadratic functional depending on the great number N of binary variables s_i :

$$E(\mathbf{s}) = -\frac{(\mathbf{Js}, \mathbf{s})}{2N} = -\frac{1}{2N} \sum_{i,j=1}^N J_{ij} s_i s_j \longrightarrow \min, \quad s_i = \pm 1. \quad (1)$$

This problem arises in a lot of scientific fields of knowledge beginning from physics of magnetic materials and neural networks up to analysis of results of physical experiments and logistics. Usually the connection matrix $\mathbf{J} = (J_{ij})_1^N$ is supposed to be symmetric one with zero diagonal elements: $J_{ij} = J_{ji}$, $J_{ii} = 0$. The state of the system as a whole is given by N -dimensional vector $\mathbf{s} = (s_1, s_2, \dots, s_N)$. Such vectors will be called *configuration vectors* or simply *configurations*. The characteristic $E(\mathbf{s})$ that has to be minimized will be called *the energy* of the state, and the configuration providing the global minimum of the functional (1) will be called *the ground state*.

In general the number of local minima of the functional (1) is exponentially large. Practically all minimization algorithms guarantee finding of a local minimum only. The exceptions are very rare and, as a rule, they are relied on specific properties of the connection matrix [1], [2]. The most widespread is the *random minimization* [3]. According this algorithm the spin dynamics is started from a random configuration. In randomized order the states of dissatisfied spins are changed. As a result the dynamic system step by step falls into the nearest local minimum. We used just the random minimization in our computer simulations (Section 3).

Very little is known about properties of the energy surface of the functional (1), namely, about the number and the structure of the set of local minima, about the ground

state and the probability to find it and so on. In fact there is only one nontrivial connection matrix for which the ground state of the functional (1) can be indicated exactly. This is the Hebb matrix in the case when the value of the loading parameter $\alpha = M/N$ is small: $\alpha < 0.07$ [3]. Then the global minimum of the functional (1) is achieved at any of M random patterns.

Due to discrete character of the problem its theoretical analysis is very rare. From recent results let us point out the papers [4], [5], where the authors succeeded in connecting the depth of the local minimum with the probability of its random finding, and also described some characteristics of the energy surface.

In our work we introduce a class of *additive matrices* whose elements are pair-wise sums of a set of predetermined numbers a_i :

$$J_{ij} = (1 - \delta_{ij})(a_i + a_j), \quad i, j = 1, \dots, N, \quad \text{where } \{a_i\}_1^N \in \mathbf{R}^1 \quad (2)$$

and δ_{ij} is the Kronecker delta symbol.

The additive matrices generalize a special class of Hebb's matrices analyzed in [6]. For the functional (1) with the connection matrix (2) the ground state can be obtained exactly. We succeeded in presentation additive matrices in the form when the dependence of the ground state on external parameters of the problem can be described analytically. When the ground state is known, interesting results can be obtained with the aid of computer simulation. In the next Section we present the theory relating to the problem. In Section 3 we give the results of computer simulations.

2 The Ground State

1. It can be verified directly that for an additive matrix (2) the value of the functional (1) is equal to

$$E(\mathbf{s}) = \frac{(\mathbf{e}, \mathbf{s}) - (\mathbf{s}, \mathbf{e})(\mathbf{s}, \mathbf{a})}{N}. \quad (3)$$

Here $\mathbf{a} = (a_1, a_2, \dots, a_N)$ is N -dimensional vector whose coordinates are a_i , and $\mathbf{e} = (1, 1, \dots, 1)$ is the "bisector" of the principal orthant of the space \mathbf{R}^N . From minimization of the functional (3) one can pass to *maximization* of the functional

$$F(\mathbf{s}) = (\mathbf{s}, \mathbf{e})(\mathbf{s}, \mathbf{a}) \longrightarrow \max. \quad (4)$$

Let us denote by Σ_k the class of all configurations \mathbf{s} for which exactly k coordinates are equal "-1":

$$\Sigma_k = \{\mathbf{s} : (\mathbf{s}, \mathbf{e}) = N - 2k\}, \quad k = 0, 1, \dots, N.$$

The class Σ_k consists of C_k^N configurations. For all these configurations the first multiplier in the expression (4) takes the same value $N - 2k$. Consequently, to maximize (4) among configurations from the class Σ_k , it is sufficient to find a vector $\mathbf{s} \in \Sigma_k$ maximizing the scalar product (\mathbf{s}, \mathbf{a}) . This problem is not so difficult (see item 3).

2. Suppose, we can find the vector \mathbf{s} maximizing the scalar product (\mathbf{s}, \mathbf{a}) in the class Σ_k . Let us denote this vector as $\mathbf{s}(k)$, and let the value of the functional (4) for this vector be $F_k = F(\mathbf{s}(k))$:

$$F_k = (N - 2k)(\mathbf{s}(k), \mathbf{a}) = \max_{\mathbf{s} \in \Sigma_k} F(\mathbf{s}), \quad k = 0, 1, \dots, N.$$

When finding all these vectors $\mathbf{s}(k)$ ($k = 0, 1, \dots, N$), it is easy to find the global maximum of the functional (4), since the functional reaches its maximal value on one of the vectors $\mathbf{s}(k)$.

Note we do not need to compare between themselves all $N + 1$ numbers F_k , but the first half of them only. The reason is that for any k the classes Σ_k and Σ_{N-k} are inversion of each other: $\Sigma_{N-k} = -\Sigma_k$. Since for any configuration \mathbf{s} the equality $F(\mathbf{s}) = F(-\mathbf{s})$ is fulfilled, we obtain that $F_k = F_{N-k}$ for all values of k . Combining the cases of even and odd N in one formula we obtain that to find the global maximum of the functional (4) it is necessary to find the largest of the values F_k , when $k \leq n = [N/2]$:

$$F_0, F_1, \dots, F_n, \quad n = \left[\frac{N}{2} \right].$$

3. Without loss of generality the numbers $\{a_i\}_1^N$ can be put in order according their increase:

$$a_1 < a_2 < \dots < a_N. \quad (5)$$

Let us take any $k \leq n$. It is easy to see that the scalar product (\mathbf{s}, \mathbf{a}) reaches its maximum inside the class Σ_k when the configuration vector is

$$\mathbf{s}(k) = (\underbrace{-1, -1, \dots, -1}_k, 1, 1, \dots, 1). \quad (6)$$

Indeed,

$$(\mathbf{s}(k), \mathbf{a}) = - \sum_{i=1}^k a_i + \sum_{i=k+1}^N a_i = \hat{a} - 2\hat{a}_k,$$

where

$$\hat{a} = \sum_{i=1}^N a_i, \quad \hat{a}_k = \sum_{i=1}^k a_i, \quad \text{and } \hat{a}_0 = 0. \quad (7)$$

Let \mathbf{s} be another configuration vector from the class Σ_k for which numbers of negative coordinates $j_1 < j_2 < \dots < j_k$ dose not take the first places. The scalar product (\mathbf{s}, \mathbf{a}) is equal to $(\mathbf{s}, \mathbf{a}) = \hat{a} - 2 \sum_{i=1}^k a_{j_i}$, and inequality $(\mathbf{s}(k), \mathbf{a}) > (\mathbf{s}, \mathbf{a})$ is fulfilled since $\sum_{i=1}^k a_i < \sum_{i=1}^k a_{j_i}$ for any set of indices $\{j_1, j_2, \dots, j_k\}$ that differs from $\{1, 2, \dots, k\}$.

Thus, under the condition of ordering (5), to find the global minimum of the functional (3) it is necessary to find the largest among the numbers

$$F_k = (N - 2k)(\hat{a} - 2\hat{a}_k), \quad k = 0, 1, \dots, n = \left[\frac{N}{2} \right], \quad (8)$$

where \hat{a} and \hat{a}_k are given in Eq.(7).

4. The initial problem (1)-(2) can be considered as solved: the expressions (6) restrict the set of configurations among which the ground state of the functional (2) should be found. To define which configuration is the ground state it is necessary to calculate n numbers (8) and find the largest among them. It reminds unclear under which conditions this or that configuration (6) would be the ground state. If any of them will be the ground state or not? It turned out that these questions can be answered.

Without loss of generality let us suppose that the numbers a_i have a special form:

$$a_i = \alpha_i - t, \quad \alpha_i \in [0, 1], \quad t \geq 0. \quad (9)$$

In this presentation the values α_i are positive numbers from the unit interval, and the positive parameter t can take an arbitrary value. It is not difficult to see that from the point of view of minimization of our functional an arbitrary set of numbers a_i can be reduced to the form (9). For example, let us suppose that $a_1 < \dots < a_N < 0$ and $a_N - a_1 \leq 1$. Then we set $\alpha_i = a_i - a_1$ and $t = |a_1|$. This means that the numbers a_i have the form (9). On the contrary, let the initial numbers \tilde{a}_i have different signs and take on arbitrary values: $\tilde{a}_1 < \dots < \tilde{a}_N$, and $\tilde{a} = \max(|\tilde{a}_1|, \tilde{a}_N) \gg 1$. Let us normalize these numbers dividing them by $2\tilde{a}$: $a_i = \tilde{a}_i / 2\tilde{a} \in [-1/2, +1/2]$. It is clear that the solution of the problem (1) is the same when we use initial numbers \tilde{a}_i or normalized numbers a_i . The last numbers can be presented in the form (9), if we set $\alpha_i = a_i + \frac{1}{2}$ and $t = \frac{1}{2}$. From our argumentation it follows that the numbers a_i can always be presented in the form (9). Then the following statement is right (the proof see in the Appendix).

Theorem. When t increasing from the initial value $t = 0$, the ground state sequentially coincides with the vectors $s(k)$ (6) in the following order:

$$s(0) \rightarrow s(1) \rightarrow \dots \rightarrow s(k-1) \rightarrow s(k) \rightarrow \dots \rightarrow s(n-1) \rightarrow s(n). \quad (10)$$

The jump of the ground state $s(k-1) \rightarrow s(k)$ occurs when t transfers through the critical value:

$$t_k = \frac{\hat{\alpha} - 2\hat{\alpha}_k + (N - 2k + 2)\alpha_k}{2(N - 2k + 1)}, \quad k = 1, 2, \dots, n, \quad (11)$$

where analogously of Eq.(7) $\hat{\alpha} = \sum_{i=1}^N \alpha_i$ and $\hat{\alpha}_k = \sum_{i=1}^k \alpha_i$. When t belongs to the interval $[t_k, t_{k+1}]$, the ground state of the functional is the configuration $s(k)$.

This theorem generalizes the previous results obtained in [6]. The theorem describes exhaustively the behavior of the ground state for the problem (1)-(2). Depending on the values of external parameters $\{a_i\}$ each of the configurations $s(k)$, $k = 0, 1, \dots, n$ can turn out to be the ground state of the functional. For t from the interval $[t_k, t_{k+1}]$ the energy of the ground state is the linear function of the parameter t . It can be easily seen from the expressions (8) substituting the values a_i in the form (9):

$$E_k(t) = A_k + t \cdot B_k, \quad k = 0, 1, \dots, n, \quad (12)$$

where up to the factor $1/2N$ we have:

$$A_k = \hat{\alpha} - (N - 2k)(\hat{\alpha} - 2\hat{\alpha}_k), \quad B_k = (N - 2k)^2 - N. \quad (13)$$

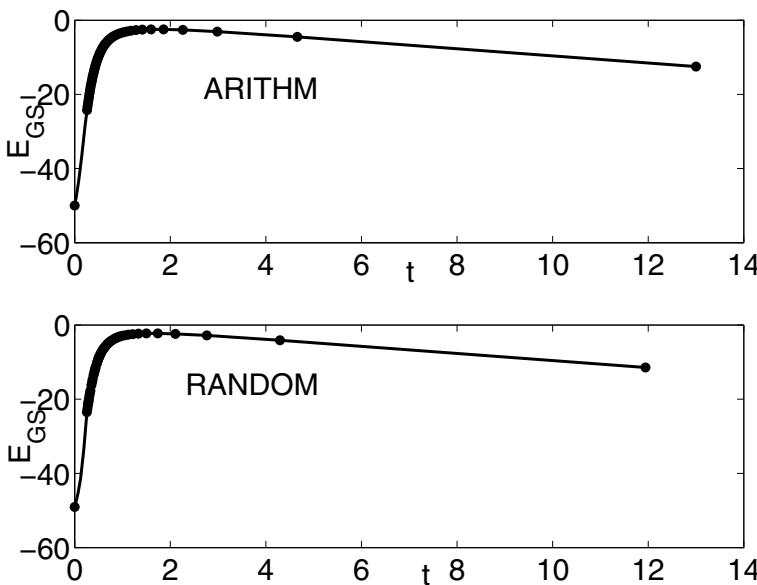


Fig. 1. The dependence of the ground state energy E_{GS} on the parameter t for additive matrices of the dimensionality $N = 100$: the upper one is the arithmetical additive matrix, and the lower one is the random matrix (see the body of the text). The values E_{GS} for the points t_k are marked.

In Fig.1 for $N = 100$ it is shown how the energy of the ground state depends on the parameter t . The upper panel corresponds to the case when the values α_i constitute the arithmetical progression: $\alpha_i = i/N$, $i = 1, 2, \dots, N$. On the lower panel the analogous plot is shown for a random additive matrix, when α_i are random numbers from the interval $[0, 1]$. Along the abscissa axis the values of the parameter t are shown, along the axis of ordinates we show the energy of the ground state calculated in the points t_k (11). The first value of the parameter t for which the energy of the ground state is calculated is equal to zero: $t_0 = 0$. Since both plots are very similar, we analyze only one of them; for example the upper one.

We see that the energy of the ground state is nontrivially depended on the parameter t . For small values, $t \sim 0$, very deep minima correspond to the ground state. Then, when t increases, the depth of the global minimum decreases very quickly and it reaches a minimal value when $t \approx 2$. For these values of t all matrix elements become negative. During further increase of t the depth of the global minimum slowly but steadily increases. It becomes deeper and deeper. Which properties of the energy surface reflect non-monotone change of the depth of the global minimum? What properties are responsible for its minimal depth? For the time being we cannot answer these questions. Using formulae (10)-(13) everyone can be certain of universal character of the curves shown in Fig.1.

Up till now we can neither extend these results onto local minima of the functional, nor obtain analytical description of other interesting characteristics such as the number of different minima, distribution of local minima with respect to their depths and

distances to the ground state and so on. However, if the ground state is known, these characteristics can be studied with the aid of computer simulations. Now we turn to presentation of these results.

3 Computer Simulation

For given N and $\{\alpha_i\}$ for each value of t we generated an additive matrix. We did 10^5 random starts (see Introduction) and obtained the same number of local minima. For each minimum we fixed its depth (the energy E_l), the relative Hamming distance D_l between the minimum and the ground state and other characteristics. Thus as a result of a great number of random trials for each value of t we could estimate: a) the probability of random finding of the ground state p_{GS} ; b) the deepest of the obtained minimum and the distance from it to the ground state; c) the number of different minima K , their distribution over energies and distances from the ground state and so on. The parameter t was varied from zero up to the maximal value t_n . For two dimensionalities $N = 100$ and $N = 1000$ such experiments were done for both arithmetical and random additive matrices.

In Fig. 2 for the arithmetical additive matrix of dimensionality $N = 100$ the dependence of some of the listed characteristics on the parameter t is shown. Let us explain what the graphs shown on different panels of the figure mean.

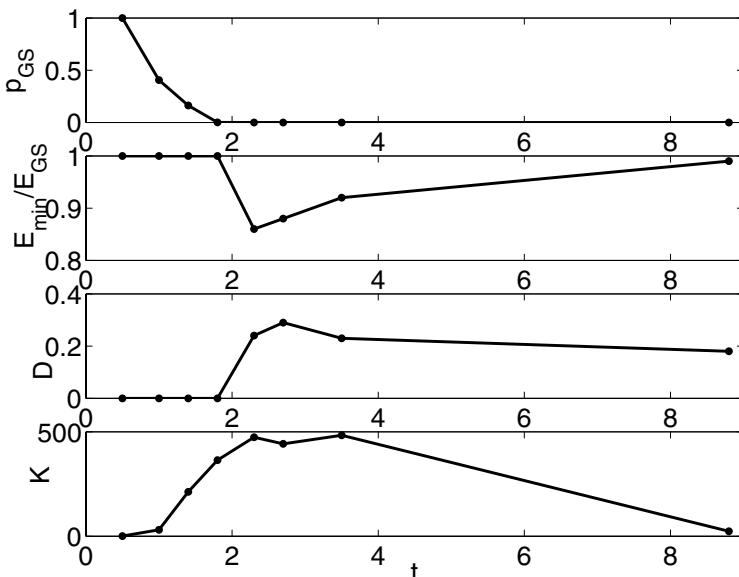


Fig. 2. For arithmetical additive matrix of dimensionality $N = 100$ the following graphs are shown: on the upper panel is the probability to find the ground state; on the next panel is the ratio of depth of the deepest found minimum to the depth of the global minimum; on the next panel is the relative Hamming distance between the deepest minimum and the ground state; on the bottom panel is the number of different energies of local minima

On the Upper Panel. The probability to find the ground state p_{GS} is shown. We see that in the region of small values of t ($t < 1.8$), where the depth of the global minimum is large, the probability to find the ground state is notably different from zero. On the contrary, in the region of large values of t , where the global minimum becomes rather shallow, the probability to find it is equal to zero (it is less than 10^{-5}). At the same time it is not important which configuration $\mathbf{s}(k)$ is the ground state.

Apparently, such behavior of the probability p_{GS} is one more confirmation of the law, which was theoretically predicted in [3], [4]: the deeper minimum, the greater probability to find it under the random search.

For the matrix of dimensionality $N = 1000$ the behavior of the given characteristic is an analogous one. The value of the parameter t for which the probability to find the ground state becomes zero, increases up to the value $t \approx 4$.

On the Second Panel from the Top. The ratio of the found deepest minimum E_{\min} to the global minimum E_{GS} , E_{\min}/E_{GS} , is shown. This ratio takes on a value from the interval $[0, 1]$. At first, while the ground state still can be found, this ratio is equal to 1. Then in the region of the values $t \approx 2.3 - 2.5$ this characteristic has a sharp downward excursion, which soon changes to a steady increasing and tends to 1 asymptotically. The minimal value of this characteristic is $E_{\min}/E_{GS} \approx 0.85$. It shows that in the worst case the objective function is 15% less than the optimal value.

For matrices of dimensionality $N = 1000$ the behavior of the ratio E_{\min}/E_{GS} is absolutely analogous. The deepest downward excursion of the graph takes place when $t \approx 4$, and its depth increases noticeably: the minimal value of the ratio is equal $E_{\min}/E_{GS} \approx 0.5$. In other words, when the dimensionality of the problem increases the found suboptimal solution will be worse comparing with the global minimum.

Note, that for the large values of $t \sim 7 - 8$, when the ratio E_{\min}/E_{GS} is close to 1, the probability to find the ground state as before is equal to 0. The same also takes place in the case $N = 1000$.

On the Second Panel from the Bottom. It is shown how the distance D between the deepest local minimum and the ground state depends on the parameter t . (By the distance we understand the relative Hamming distance $D = (N - \text{abs}(\mathbf{s}, \mathbf{s}'))/2N \in [0, 0.5]$.)

At first, while the ground state still can be found this distance is equal to 0 (see the beginning of the graph). Then in the interval of the "worst" values of t the distance D increases sharply up to the value $D_{\max} \approx 0.3$. After that the distance between the deepest local minimum and the ground state is stabilized near the value $D = 0.2$. Let us add that for additive matrices of dimensionality $N = 1000$ suboptimal solution is far away from the ground state. This distance is $D \approx 0.4$.

The general conclusion is as follows: for rather large values of t , when as a result of the random search it is possible to find suboptimal solution only, this solution is sufficiently far from the ground state. However, the ratio of the minima depths E_{\min}/E_{GS} can be of order of 1 (in Fig.2 this situation corresponds to the values of $t > 4$). This combination of properties is possible only if the energy surface consists of a large number of local minima, which depths not strongly differ one from each other and from the global minimum. We may conclude, that for large values of t , when elements of

connection matrix are large negative numbers, the construction of the energy surface is as aforesaid.

On the Bottom Panel. We show the dependence of the number of different energies of local minima K on the value of the parameter t . As a rule each energy is many times degenerated. To estimate the number of different local minima it is necessary to analyze how many different configurations correspond to the same energy. Nevertheless, such characteristic as the number of energy levels is also of interest.

For an arithmetical additive matrix of the dimensionality $N = 100$ the maximal value of the characteristic K is reached in the region $t \sim 3$. This maximum is comparatively small, ~ 500 . However, it turns out that each energy is many times degenerated, and the number of different local minima is an orders of magnitude greater. For a random additive matrix of the same dimensionality the maximal value of the characteristic K is equal to tens of thousands (the graph is not presented).

For the additive matrices of the dimensionality $N = 1000$ the general form of the graph of the characteristic K is analogous. In this case the maximal value, $K_{\max} \sim 4 \cdot 10^4$, is reached in the region $t \approx 5$. Since for each t only 10^5 random starts have been done, this means that each second start leads the system into new local minimum. In other words, for middle values of t the number of local minima is very big.

4 Discussion and Conclusions

For additive matrices the method of finding of the global minimum of the quadratic binary functional is pointed out. We propose the t -parametrization of additive matrices that allows one to get an exhaustive classification for all variants possible for the ground state.

For not great values of t (let us say for $t \in [0, 2]$) among matrix elements there are positive as well as negative ones; or all elements are negative, but they are small in modulus. In this case the depth of the global minimum is very big. Here the probability to find the ground state in random search is rather high: $p_{GS} \sim 0.5 - 1.0$. It can be supposed that in this case the energy surface has a small number of local minima whose depths noticeably less than the depth of the global minimum.

On the contrary, for the great values of t all matrix elements are negative and they are big in modulus. In this case it is practically impossible to find the ground state with the aid of the random minimization, since the probability to get into the global minimum is negligible small. Apparently in this case the energy surface contains very large number of local minima that only slightly differ from each other in depths. Here the global minimum is only insignificantly deeper than local minima. Varying the value of the parameter t it is possible to get over from one type of the energy surface to the other one. So, additive matrices are good models for examining the energy surfaces in the general case.

By this time additive matrices for large values of t can be used for testing of new algorithms of the quadratic binary minimization. Indeed, on the one hand, with the aid of the formulae (10)-(13) the ground state always can be found. On the other hand, for large values of the parameter t it is practically impossible to find the ground state with the aid of the random minimization.

Acknowledgment

The work has been done with participation of Anton Pichugin in the framework of the program supported by the grant #356.2008.9 of President of Russian Federation and in part by Russian Basic Research Foundation (grant #09-07-00159).

References

1. Hartmann, A.K., Rieger, H. (eds.): New optimization algorithms in physics. Wiley-VCH, Berlin (2004)
2. Junger, M.: The Spin Glass Ground State Server,
http://www.informatik.uni-koeln.de/ls_juenger/research/sgs/sgs.html
3. Hertz, J., Krogh, A., Palmer, R.: Introduction to the Theory of Neural Computation. Addison-Wesley, NY (1991)
4. Kryzhanovsky, B.V., Magomedov, B.M., Mikaelyan, A.L.: A Relation between the Depth of a Local Minimum and the Probability of Its Detection in the Generalized Hopfield Model. Doklady Mathematics 72(3), 986–990 (2005)
5. Kryzhanovsky, B.V., Kryzhanovsky, V.M., Mikaelian, A.L.: Binary optimization: A relation between the depth of a local minimum and the probability of its detection. In: 4th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2007, Anger, France, pp. 5–10 (2007)
6. Litinskii, L.B.: High-symmetry Hopfield-type neural networks. Theoretical and Mathematical Physics 118(1), 107–127 (1999)

Appendix

In the beginning of Section 2 it was shown that only one of configuration vectors $\mathbf{s}(k)$ (6), $k = 0, 1, \dots, n = [N/2]$ can be the ground state. Using the representation (9) of a_i it is easy to obtain Eq. (12) for the energies of $\mathbf{s}(k)$ -configurations: $E_k(t) = A_k + tB_k$, where A_k and B_k are given by Eq.(13). As functions of the parameter t energies $E_k(t)$ are straight lines. We have to analyze the behavior of the set $\{E_k(t)\}_0^n$. When a straight line $E_l(t)$ is lower all other straight lines, the configuration $\mathbf{s}(l)$ is the ground state.

For simplicity we restrict ourselves to the case of even $N = 2n$. Let us write down the expression (13) in more details:

$$\begin{aligned} A_0 &= -(N-1)\hat{\alpha} < A_1 < \dots < A_n = \hat{\alpha}, \\ B_0 &= (N-1)N > B_1 > \dots > B_n = -N. \end{aligned} \quad (A1)$$

When k increasing, the free term A_k of the straight line $E_k(t)$ increases monotonically. In other words, the intersection of the straight line with ordinate axis rises higher and higher. On the other hand, when k increasing the coefficient B_k decreases monotonically, so that in the end it even becomes negative. For the case $N = 6$ the typical behavior of the set of straight lines $\{E_k(t)\}_0^n$ is shown in Fig.3. We use this figure to explain how the ground state depends on the parameter t .

When $t = 0$ all the matrix elements are positive and the configuration $\mathbf{s}(0) = (1, \dots, 1)$ is the ground state. Let us increase t little by little. At first the straight line $E_0(t)$ is lower

than all other straight lines. Consequently, $s(0)$ remains the ground state. Then for some value of the parameter t the straight line $E_0(t)$ is intersected by another straight line. After that this straight line turns out to be lower than all other straight lines. Taking into account the relations (A1) it is easy to see that the first straight line that intersects $E_0(t)$ is $E_1(t)$ (see also Fig.3). After this intersection the configuration $s(1)$ becomes the ground state. It is the ground state until another straight line intersects the straight line $E_1(t)$. After that this straight line turns out to be lower than all other straight lines. From the aforesaid argumentation it is evident that it will be the straight line $E_2(t)$ (see Fig.3). Then the configuration $s(2)$ will be the ground state, and so on. It can be shown that if the straight line $E_{k-1}(t)$ is lower than all other straight lines, the first straight line that intersects $E_{k-1}(t)$ is $E_k(t)$. The intersection takes place in the point t_k (11) that is the solution of equation $A_{k-1} + t \cdot B_{k-1} = A_k + t \cdot B_k$.

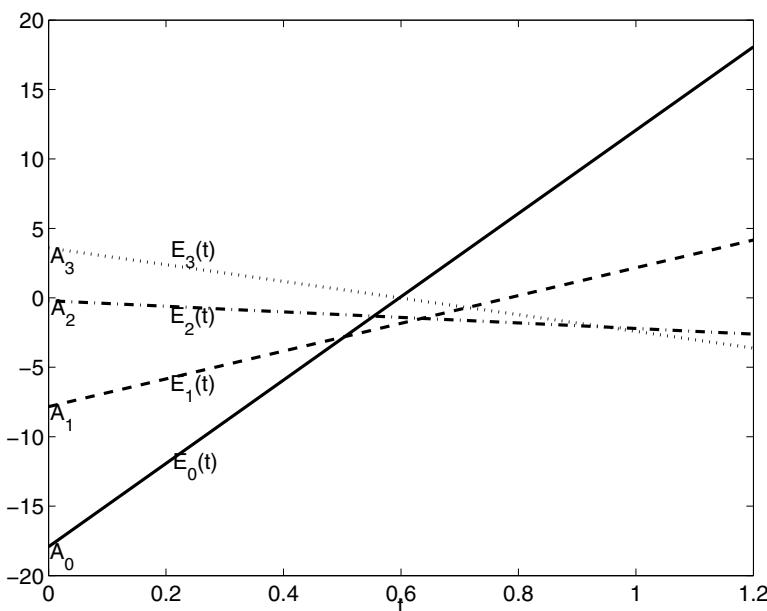


Fig. 3. For random additive matrix of dimensionality $N = 6$ the straight lines $E_k(t) = A_k + t \cdot B_k$ are shown for $k = 0, 1, 2, 3$ (see the body of the text)

Mutual Learning with Many Linear Perceptrons: On-Line Learning Theory

Kazuyuki Hara¹, Yoichi Nakayama², Seiji Miyoshi³, and Masato Okada^{4,5}

¹ Tokyo Metropolitan College of Industrial Technology, 1-10-40, Higashi-oi,
Shinagawa Tokyo 140-0011, Japan
hara@s.metro-cit.ac.jp

² Tokyo Metropolitan College of Technology, 1-10-40, Higashi-oi,
Shinagawa Tokyo 140-0011, Japan

³ Faculty of Engineering Science, Kansai University,
3-3-35, Yamate-cho, Suita, Osaka, 564-8680, Japan
miyoshi@ipcku.kansai-u.ac.jp

⁴ Graduate School of Frontier Sciences, The University of Tokyo,
5-1-5, Kashiwanoha, Kashiwa-shi, Chiba, 277-8561, Japan
okada@k.u-tokyo.ac.jp

⁵ Brain Science Institute, Riken, 2-1, Hirosawa, Wako, Saitama, 351-0198, Japan

Abstract. We propose a new mutual learning using many weak learner (or student) which converges into the identical state of Bagging that is kind of ensemble learning, within the framework of on-line learning, and have analyzed its asymptotic property through the statistical mechanics method. Mutual learning involving more than three students is essential compares to two student case from a viewpoint of variety of selection of a student acting as teacher. The proposed model consists of two learning steps: many students independently learn from a teacher, and then the students learn from others through the mutual learning. In mutual learning, students learn from other students and the generalization error is improved even if the teacher has not taken part in the mutual learning. We demonstrate that the learning style of selecting a student to act as teacher randomly is superior to that of cyclic order by using principle component analysis.

1 Introduction

As a model incorporating the interaction between students, Kinzel proposed mutual learning within the framework of on-line learning[12]. Kinzel's model employs two students, and one student learns with the other student acting as a teacher. The target of his model is to obtain the identical networks through such learning. On the other hand, ensemble learning algorithms, such as bagging[3] and Ada-boost[4], try to improve upon the performance of a weak learning machine by using many weak learning machines; such learning algorithms have recently received considerable attention. We have noted, however, that the mechanism of integrating the outputs of many weak learners in ensemble learning is similar to that of obtaining the identical networks through mutual learning.

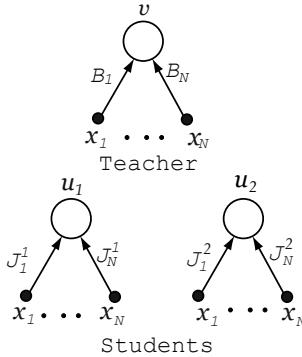


Fig. 1. Network structure of latent teacher and student networks, all having the same network structure

With regard to the learning problem, how the student approaches the teacher is important. However, Kinzel^[12] does not deal with the teacher-student relation since a teacher is not employed in his model. In contrast to Kinzel's model, we have proposed mutual learning between two students who learn from a teacher in advance^[5,6]. In our previous work^[5,6], we showed that the generalization error of the students becomes smaller through the mutual learning even if the teacher does not take part in the mutual learning. Our previous work^[5,6] treated a special case where the number of students is two. This paper treats the general case where the number of students is arbitrary. When the number of students becomes general, additional degrees of freedom associated with the selection of learning order are generated, and the problem settings become essentially different from the two students case. We formulate a new mutual learning algorithm, and then we analyze the asymptotic property of the proposed learning algorithm through statistical mechanics.

2 Formulation of Mutual Learning with a Latent Teacher

In this section, we formulate the latent teacher and student networks, and the mutual learning algorithms. We assume the latent teacher and student networks receive N -dimensional input $\mathbf{x}(m) = (x_1(m), \dots, x_N(m))$ at the m -th learning iteration as shown in Fig. 1. Learning iteration m is ignored in the figure.

The latent teacher network is a linear perceptron, and the student networks are K linear perceptrons. We also assume that the elements $x_i(m)$ of the independently drawn input $\mathbf{x}(m)$ are uncorrelated random variables with zero mean and $1/N$ variance; that is, the elements are drawn from a probability distribution $P(\mathbf{x})$. In this paper, the thermodynamic limit of $N \rightarrow \infty$ is assumed. Thermodynamic limit means that for the limit of system size N to be infinity, the law of large numbers and the central limit theorem are effected. We can then depict the system behavior by using a small number of parameters. At the limit, the size of input vector $\|\mathbf{x}\|$ then becomes one.

$$\langle x_i \rangle = 0, \quad \langle (x_i)^2 \rangle = \frac{1}{N}, \quad \|\mathbf{x}\| = 1, \quad (1)$$

where $\langle \dots \rangle$ denotes average, and $\|\cdot\|$ denotes the norm of a vector.

The latent teacher network is a linear perceptron, and is not subject to training. Thus, the weight vector is fixed in the learning process. The output of the latent teacher $v(m)$ for N -dimensional input $\mathbf{x}(m)$ at the m -th learning iteration is

$$v(m) = \sum_{i=1}^N B_i x_i(m) = \mathbf{B} \cdot \mathbf{x}(m), \quad (2)$$

$$\mathbf{B} = (B_1, B_2, \dots, B_N), \quad (3)$$

where latent teacher weight vector \mathbf{B} is an N -dimensional vector like the input vector, and each element B_i of the latent teacher weight vector \mathbf{B} is drawn from a probability distribution of zero mean and unit variance. Assuming the thermodynamic limit of $N \rightarrow \infty$, the size of latent teacher weight vector $\|\mathbf{B}\|$ becomes \sqrt{N} .

$$\langle B_i \rangle = 0, \quad \langle (B_i)^2 \rangle = 1, \quad \|\mathbf{B}\| = \sqrt{N}. \quad (4)$$

The output distribution of the latent teacher $P(v)$ follows a Gaussian distribution of zero mean and unit variance in the thermodynamic limit of $N \rightarrow \infty$.

The K linear perceptrons are used as student networks that compose the mutual learning machine. Each student network has the same architecture as the latent teacher network. For the sake of analysis, we assume that each element of $\mathbf{J}^k(0)$ which is the initial value of the k -th student weight vector \mathbf{J}^k is drawn from a probability distribution of zero mean and unit variance. The norm of the initial student weight vector $\|\mathbf{J}^k(0)\|$ is \sqrt{N} in the thermodynamic limit of $N \rightarrow \infty$,

$$\langle J_i^k(0) \rangle = 0, \quad \langle (J_i^k(0))^2 \rangle = 1, \quad \|\mathbf{J}^k(0)\| = \sqrt{N}. \quad (5)$$

The k -th student output $u_k(m)$ for the N -dimensional input $\mathbf{x}(m)$ at the m -th learning iteration is

$$u_k(m) = \sum_{i=1}^N J_i^k(m) x_i(m) = \mathbf{J}^k(m) \cdot \mathbf{x}(m), \quad (6)$$

$$\mathbf{J}^k(m) = (J_1^k(m), J_2^k(m), \dots, J_N^k(m)). \quad (7)$$

Generally, the norm of student weight vector $\|\mathbf{J}^k(m)\|$ changes as the time step proceeds. Therefore, the ratio l_k of the norm to \sqrt{N} is considered and is called the length of student weight vector \mathbf{J}^k . The norm at the m -th iteration is $l_k(m)\sqrt{N}$, and the size of $l_k(m)$ is $O(1)$.

$$\|\mathbf{J}^k(m)\| = l_k(m)\sqrt{N}. \quad (8)$$

The distribution of the output of the k -th student $P(u_k)$ follows a Gaussian distribution of zero mean and l_k^2 variance in the thermodynamic limit of $N \rightarrow \infty$.

Next, we formulate the learning algorithm. After the students learn from a latent teacher, mutual learning is carried out [5]. The learning equation of the mutual learning is

$$\mathbf{J}^k(m+1) = \mathbf{J}^k(m) + \eta_k \left(u_{k'} - u_k \right) \mathbf{x}(m) \quad (9)$$

Here, k is a student and k' is a student to act as a teacher. m denotes the iteration number. We use the gradient descent algorithm in this paper, while another algorithm was used in Kinzel's work [1]. Equation (9) shows that mutual learning is carried out between two students. Therefore, the teacher used in the initial learning is called a latent teacher.

In the mutual learning, selection of a student to act as a teacher is important. In this paper, a student to act as a teacher is selected at random from all the students, then only the statistical effects is learned by a student and therefore a student tend to learn the average of all the students. Keeping this in mind, the learning equation is rewritten by the next equation.

$$\begin{aligned} \mathbf{J}^k(m+1) &= \mathbf{J}^k(m) + \eta_k \left(\frac{1}{K} \sum_{i=1}^K u_i(m) - u_k(m) \right) \mathbf{x}(m) \\ &= \mathbf{J}^k(m) + \eta_k (\bar{u}(m) - u_k(m)) \mathbf{x}. \end{aligned} \quad (10)$$

Here, \bar{u} is average of the student outputs and is to act as a teacher.

When the interaction between students is introduced, the performance of students may be improved if they exchange knowledge that each student has acquired from the latent teacher in the initial learning. In other words, two students approach each other through mutual learning, and tend to move towards the middle of the initial weight vectors. This tendency is similar to the integration mechanism of Bagging, so mutual learning may mimic this mechanism.

3 Theory

In this section, we first derive the differential equations of two order parameters which depict the behavior of mutual learning. After that, we derive an auxiliary order parameter which depicts the relationship between the teacher and students. We then rewrite the generalization error using these order parameters. We first derive the differential equation of the length of the student weight vector l_k . l_k is the first order parameter of the system. We modify the length of the student weight vector in Eq. (8) as $\mathbf{J}^k \cdot \mathbf{J}^k = N l_k^2$. To obtain a time dependent differential equation of l_k , we square both sides of Eq. (10). We then average the term of the equation using the distribution of $P(u_k, u_{k'})$. Note that \mathbf{x} and \mathbf{J}^k are random variables, so the equation becomes a random recurrence formula. We formulate the size of the weight vectors to be $O(N)$, and the size of input \mathbf{x} is $O(1)$, so the length of the student weight vector has a self-averaging property. Here, we rewrite m as $m = Nt$, and represent the learning process using continuous time

t in the thermodynamic limit of $N \rightarrow \infty$. We then obtain the deterministic differential equation of l_k ,

$$\frac{dl_k^2}{dt} = \frac{2\eta_k}{K} \left(\sum_{i \neq k}^K Q_{ik} - (K-1)l_k^2 \right) + \frac{\eta_k^2}{K^2} \left\{ (K-1)^2 l_k^2 + \sum_{i \neq k}^K K(l_i^2 - 2KQ_{ik}) + 2 \sum_{i=1}^{K-1} \sum_{j>i}^K Q_{ij} \right\}. \quad (11)$$

Here, $k = 1 \sim K$. $Q_{kk'} = q_{kk'} l_k l_{k'}$, and $q_{kk'}$ is the overlap between \mathbf{J}^k and $\mathbf{J}^{k'}$, defined as

$$q_{kk'} = \frac{\mathbf{J}_k \cdot \mathbf{J}_{k'}}{|\mathbf{J}^k| |\mathbf{J}^{k'}|} = \frac{\mathbf{J}^k \cdot \mathbf{J}^{k'}}{N l_k l_{k'}}, \quad (12)$$

$q_{kk'}$ is the second order parameter of the system. The overlap $q_{kk'}$ also has a self-averaging property, so we can derive the differential equation in the thermodynamic limit of $N \rightarrow \infty$. The differential equation is derived by calculating the product of the learning equation (eq. (9)) for \mathbf{J}^k and $\mathbf{J}^{k'}$, and we then average the term of the equation using the distribution of $P(u_k, u_{k'})$. After that, we obtain the deterministic differential equation as

$$\frac{dQ_{kk'}}{dt} = \frac{1}{K} \left\{ \left(l_k^2 + \sum_{i \neq k}^K Q_{ik} \right) (\eta_k - \eta_k \eta_{k'}) + \left(l_{k'}^2 + \sum_{i \neq k'}^K Q_{ik'} \right) (\eta_{k'} - \eta_k \eta_{k'}) \right\} + \frac{\eta_k \eta_{k'}}{K^2} \left(\sum_{i=1}^K l_i^2 + 2 \sum_{i=1}^{K-1} \sum_{j>i}^K Q_{ij} \right) - Q_{kk'} (\eta_k + \eta_{k'} - \eta_k \eta_{k'}). \quad (13)$$

Equations (11) and (13) form closed differential equations.

To depict the behavior of mutual learning with a latent teacher, we have to obtain the differential equation of overlap R_k , which is a direction cosine between latent teacher weight vector \mathbf{B} and the k -th student weight vector \mathbf{J}^k defined by eq. (14). We introduce R_k as the third order parameter of the system.

$$R_k = \frac{\mathbf{B} \cdot \mathbf{J}^k}{|\mathbf{B}| |\mathbf{J}^k|} = \frac{\mathbf{B} \cdot \mathbf{J}^k}{N l_k} \quad (14)$$

For the sake of convenience, we write the overlap between the latent teacher weight vector and the student weight vector as r_k and $r_k = R_k l_k$. The differential equation of overlap r_k is derived by calculating the product of \mathbf{B} and eq. (9), and we then average the term of the equation using the distribution of $P(v, u_k, u_{k'})$. The overlap r_k also has a self-averaging property, and in the thermodynamic

limit, the deterministic differential equation of r_k is then obtained through a calculation similar to that used for l_k .

$$\frac{dr_k}{dt} = \frac{\eta_k}{K} \left(\sum_{i \neq k}^K r_i - (K-1)r_k \right) \quad (15)$$

The squared error for the k -th student ϵ^k is then defined using the output of the latent teacher and that of the student as given in eqs. (2) and (6), respectively.

$$\epsilon^k = \frac{1}{2} (\mathbf{B} \cdot \mathbf{x} - \mathbf{J}^k \cdot \mathbf{x})^2 \quad (16)$$

The generalization error for the k -th student ϵ_g^k is given by the squared error ϵ^k in eq. (16) averaged over the possible input \mathbf{x} drawn from a Gaussian distribution $P(\mathbf{x})$ of zero mean and $1/N$ variance.

$$\epsilon_g^k = \int d\mathbf{x} P(\mathbf{x}) \epsilon^k = \frac{1}{2} \int d\mathbf{x} P(\mathbf{x}) (\mathbf{B} \cdot \mathbf{x} - \mathbf{J}^k \cdot \mathbf{x})^2. \quad (17)$$

This calculation is the N -th Gaussian integral with \mathbf{x} and it is hard to calculate. To overcome this difficulty, we employ coordinate transformation from \mathbf{x} to v and u_k in eqs. (2) and (6). Note that the distribution of the output of the students $P(u_k)$ follows a Gaussian distribution of zero mean and l_k^2 variance in the thermodynamic limit of $N \rightarrow \infty$. For the same reason, the output distribution for the latent teacher $P(v)$ follows a Gaussian distribution of zero mean and unit variance in the thermodynamic limit. Thus, the distribution $P(v, u_k)$ of latent teacher output v and the k -th student output u_k is

$$P(v, u_k) = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp \left[-\frac{(v, u_k)^T \Sigma^{-1} (v, u_k)}{2} \right], \quad (18)$$

$$\Sigma = \begin{pmatrix} 1 & r_k \\ r_k & l_k^2 \end{pmatrix}. \quad (19)$$

Here, T denotes the transpose of a vector, r_k denotes $r_k = R_k l_k$, and R_k is the overlap between the latent teacher weight vector \mathbf{B} and the student weight vector \mathbf{J}^k defined by eq. (14). Hence, by using this coordinate transformation, the generalization error in eq. (17) can be rewritten as

$$\epsilon_g^k = \frac{1}{2} \int dv du_k (v - u_k)^2 = \frac{1}{2} (1 - 2r_k + l_k^2). \quad (20)$$

Consequently, we calculate the dynamics of the generalization error by substituting the time step value of $l_k(t)$, $Q(t)$, and $r_k(t)$ into eq. (20).

4 Results

In this section, we discuss the dynamics of the order parameters and their asymptotic properties, and then discuss the relationship between mutual learning and

Bagging. For the sake of simplicity, the initial weight vectors of the students are homogeneously correlated. From the symmetry of the evolution equation for updating the weight vector, $l_k(t) = l(t)$, $Q_{kk'}(t) = Q(t)$, $r_k(t) = r(t)$ are obtained. We assume the learning step size $\eta_k = \eta$. By substitute above conditions into Eqs. (11), (13), and (15), we get

$$\frac{dl^2}{dt} = -\frac{K-1}{K}(l^2 - Q)(2\eta - \eta^2), \quad (21)$$

$$\frac{dQ}{dt} = \frac{1}{K}(l^2 - Q)(2\eta - \eta^2), \quad (22)$$

$$\frac{dr}{dt} = 0. \quad (23)$$

Here, Eqs. (21) and (22) form closed differential equations. These equations can be solved analytically.

$$l^2(t) = \frac{K-1}{K}(l^2(0) - Q(0)) \exp(-(2\eta - \eta^2)t) + \frac{l^2(0) + (K-1)Q(0)}{K}, \quad (24)$$

$$Q(t) = -\frac{1}{K}(l^2(0) - Q(0)) \exp(-(2\eta - \eta^2)t) + \frac{l^2(0) + (K-1)Q(0)}{K}, \quad (25)$$

where $l^2(0)$ is the initial value of $l^2(t)$, and $Q(0)$ is the initial value of $Q(t)$. From Eqs. (24) and (25), $l^2(t)$ and $Q(t)$ are diverged when $\eta \geq 2$, learning will not converge in this condition.

Equation (23) depicts dynamics of overlap between the teacher and the student. The analytical solution of Eq. (23) is easily given by

$$r(t) = r(0). \quad (26)$$

Here, $r(0)$ is the initial value of $r(t)$. By substituting Eqs. (24) and (26) into (20), we can rewrite the generalization error for K students.

$$\begin{aligned} \epsilon_g^K(t) &= \frac{1}{2} \left(1 - 2r(0) + \frac{K-1}{K}(l^2(0) - Q(0)) \exp(-(2\eta - \eta^2)t) \right. \\ &\quad \left. + \frac{l^2(0) + (K-1)Q(0)}{K} \right) \end{aligned} \quad (27)$$

The asymptotic property of the order parameters $l(\infty)$, $Q(\infty)$ and $r(\infty)$ is given by substituting $t \rightarrow \infty$ into Eqs. (24), (25) and (26),

$$l^2(\infty) = Q(\infty) = \frac{l^2(0) + (K-1)Q(0)}{K}, \quad (28)$$

$$r(\infty) = r(0). \quad (29)$$

Consequently, we calculate the asymptotic property of the generalization error by substituting $l_k(\infty)$, $Q(\infty)$, and $r_k(\infty)$ into eq. (20).

$$\epsilon_g(\infty)^K = \frac{1}{2} \left(1 - 2r(0) + \frac{l^2(0) + (K-1)Q(0)}{K} \right) \quad (30)$$

From Eq. (35), the generalization error of Bagging ϵ_g^B using K weak learners is given by

$$\epsilon_g^B = \frac{1}{2} \left\{ 1 - 2r + \frac{l^2 + (K-1)Q}{K} \right\}. \quad (31)$$

By substituting $l = l(0)$, $Q = Q(0)$ and $r = r(0)$ into Eq. (31), the generalization error of mutual learning ϵ_g^K was identical to the one of Bagging ϵ_g^B , then mutual learning asymptotically converged into Bagging.

Moreover, in the limit of number of students is $K \rightarrow \infty$, the generalization error is

$$\epsilon_g(\infty)^\infty \sim \frac{1}{2}(1 - 2r(0) + Q(0)). \quad (32)$$

4.1 Learning Property through Computer Simulations

Mutual learning involving more than three students is the general case, compared to the two students case, with regard to the variety of students who can be selected to act as teacher. Figure 2 shows trajectories of the student weight vectors obtained by principle component analysis (PCA) during mutual learning involving three students. (The three students are respectively referred to as A, B and C, and A teaching B is indicated as A → B.) Figure 2(a) shows results obtained through learning where one student is randomly selected to act as teacher, and for comparison (b) shows results obtained through learning in a cyclic order of A → B → C → A. The symbol "o" at the center of each figure shows the weight vector of the latent teacher B . In these figures, the horizontal axis shows the first principle component, and the vertical axis shows the second principle component. As shown, the trajectory of (a) converges with the minimum distance to the latent teacher, while that of (b) converges after a longer distance. This demonstrates that the learning style of (a) is superior to that of (b), confirming the validity of the proposed learning algorithm.

Next, we show the time dependence of the generalization error in Fig. 3. The number of students was 2, 3, 5, or 10, and the learning step size was $\eta = 0.1$. The initial conditions were $r(0) = 0.8$, $Q(0) = 0.6$, and $l(0) = 1$. The results were obtained through computer simulations with $N = 1000$ using random selection learning. In the figure, the horizontal axis is normalized time $t = m/N$, where m is the number of learning iterations. The vertical axis is the generalization error. The solid lines show the results using analytical solutions, and symbols "x", "+", "*", and "□" show the results for $K = 2, 3, 5$, and 10, respectively. We assumed a weight vector size of $O(\sqrt{N})$ and input size of $O(1)$ in the theoretical analysis. We kept these assumptions in the computer simulations, so if N is sufficiently large, the order parameters would have a self-averaging property. As shown, the

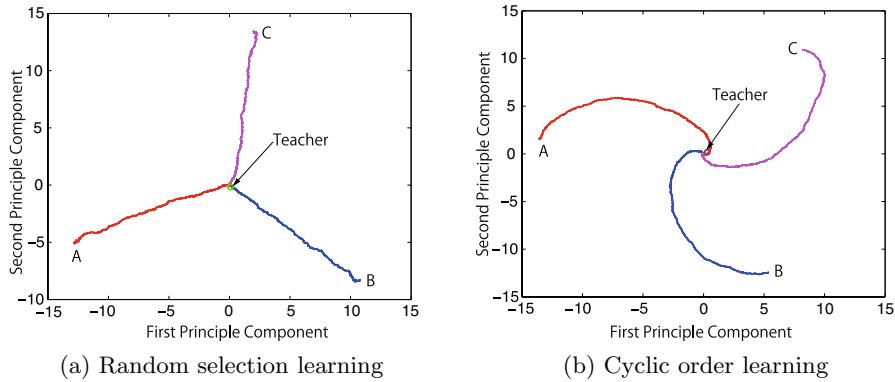


Fig. 2. Trajectory of student weight vector during learning. Three students' case.

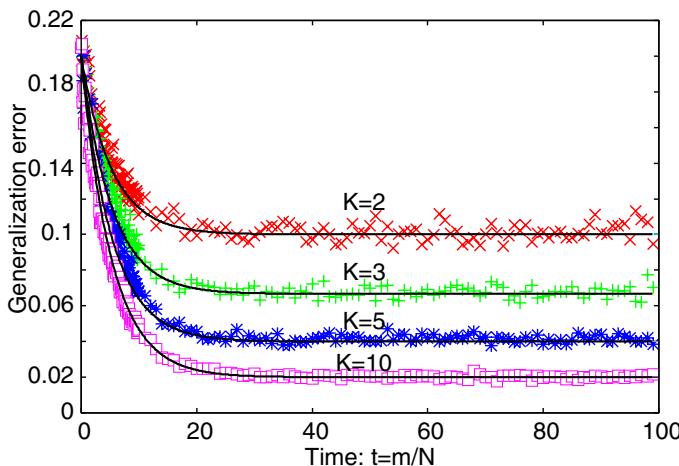


Fig. 3. Dependence of the mutual learning generalization error on the number of student networks K

analytical results agreed with those of the computer simulations, confirming the validity of the theoretical assumptions. We found that the generalization error decreased in proportion to $O(1/K)$. Moreover, the variance of the generalization error when using computer simulations tended to become smaller as the number of students K increased.

5 Conclusion

We have proposed a mutual learning algorithm using many students within the framework of on-line learning. From the results, analytical results are agreed with that of computer simulations, and the validity of the theoretical results

are shown. We showed that random selection of a student as a teacher is useful for mutual learning, and we found that the generalization error decreased in proportion to $O(1/K)$. Our future work is analyzing a mutual learning using many non-linear perceptrons.

Acknowledgment

This study has been supported by Grant-in-Aid for Scientific Research (C) No. 18500183.

References

1. Klein, E., et al.: Synchronization of neural networks by mutual learning and its application to cryptography. Proc. Neural Inf. Pro. Sys. 17, 689–696 (2004)
2. Mislovaty, R., Klein, E., Taunter, I., Kinzel, W.: Public channel cryptography by synchronization of neural networks and chaotic maps. Phys. Rev. Lett. 91(11), 118701 (2003)
3. Bran, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
4. Freund, Y., Shari'a, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55, 119–139 (1997)
5. Hara, K., Okada, M.: Statistical mechanics of mutual learning with latent teacher. J. Phys. Soc. Jpn. 76, 014001 (2007)
6. Hara, K., Yamada, T.: Optimization of the asymptotic property of mutual learning involving an integration mechanism of ensemble learning. J. Phys. Soc. Jpn. 76, 024005 (2008)

A Bagging

Bagging is a learning method using many weak learning machines to improve upon the performance of a single weak learning machine [3]. Students learn from the teacher individually, and then an ensemble output \bar{u} is calculated.

$$\bar{u} = \frac{\sum_{k=1}^K u_k}{K} = \frac{\sum_{k=1}^K (\mathbf{J}^k \cdot \mathbf{x})}{K} = \mathbf{J}^B \cdot \mathbf{x}. \quad (33)$$

The length of the weight vector l^B and the overlap r^B are given by ,

$$(l^B)^2 = \frac{l^2 + (K-1)Q}{K}, \quad r^B = \frac{1}{K} \sum_{k=1}^K r_k = r. \quad (34)$$

Here, we assumed the conditions of $l_k = l$, $Q_k = Q$, and $r_k = r$, respectively. The generalization error of ensemble output ϵ_g^B is given by substituting Eqs. (34) into Eq. (20):

$$\epsilon_g^B = \frac{1}{2} \left\{ 1 - 2r + \frac{l^2 + (K-1)Q}{K} \right\}. \quad (35)$$

Synchrony State Generation in Artificial Neural Networks with Stochastic Synapses

Karim El-Laithy and Martin Bogdan

Dept. of Computer Engineering, Faculty of Mathematics and Informatics,
University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany
{kellaithy,bogdan}@informatik.uni-leipzig.de
<http://www.informatik.uni-leipzig.de/ti/>

Abstract. In this study, the generation of temporal synchrony within an artificial neural network is examined considering a stochastic synaptic model. A network is introduced and driven by Poisson distributed trains of spikes along with white-Gaussian noise that is added to the internal synaptic activity representing the background activity (neuronal noise). A Hebbian-based learning rule for the update of synaptic parameters is introduced. Only arbitrarily selected synapses are allowed to learn, i.e. change parameter values. The average of the cross-correlation coefficients between a smoothed version of the responses of all the neurons is taken as an indicator for synchrony. Results show that a network using such a framework is able to achieve different states of synchrony via learning. Thus, the plausibility of using stochastic-based models in modeling the neural process is supported. It is also consistent with arguments claiming that synchrony is a part of the memory-recall process and copes with the accepted framework in biological neural systems.

Keywords: Neural network, temporal synchronization, stochastic synapses, neuronal states.

1 Introduction

Temporal coherence in the firing activity of groups of neurons is widely observed as a common feature throughout the neocortex [1]. The analysis of the responses of stimulated neurons of cat's visual cortex [2] confirmed that activated neurons can reliably produce synchronous discharge with the precision of a few milliseconds. Investigating the key factors in exhibiting such synchronous activity [3] related these observations to both the pure excitatory and the intrinsic time course of synaptic interactions. This coherence is believed to play an important role in neural coding and computational principles [4]. Synaptic background activity (namely the noise) was reported through the theoretical and experimental studies of synchronous activities as a key feature and it was emphasized that such background activity can affect driving coherent spiking activity [5,6,7,8].

Although the real mechanism underlying neuronal synchrony (or temporal correlation) is not completely investigated [1], the issue gained more importance

in the research since it has been viewed as a plausible solution to the "binding problem" [9][10][11] discussed in [12]. These studies argued that such neuronal temporal synchrony could allow the information about stimuli to be conveyed as temporal relations between neural sites and provide the basis for integrating different features representing the same object. Thus, binding can be defined as the ability of the biological neural system, in terms of inherited flexibilities, to construct higher level symbols by combining more elementary symbols [13].

States of synchrony are involved when any group of neurons realize a degree of synchronous activity, consequently this group of neurons exhibits a state of mental activity [2]. By entertaining such conceptual assumption and in order to observe this state, this temporal synchrony (or temporal correlation)¹ is defined over a time period T_{sync} . With this period of time (or Psychological Moment) a brain, mental or neurological state is defined. At times greater than T_{sync} one sees only a sequence of states (state history). Below this time window a state cannot be defined. The need to maintain this state of temporal correlation for periods greater than few milliseconds was supported by the argumentation in [13] in order to confine the behavioral conditions fitting the higher brain functions and difficult tasks that require sustained level of activity. Von der Malsburg stated that plausible values for this time window could be in the range of 50 - 200 millisecond and may be also extended to involve minutes if other mental aspects are in concern [12]. Within this time window, the actual signal fluctuations are not relevant (for a complete review please refer to [11][13][14]).

The generation of synchrony in artificial neural networks (ANN) is addressed in many theoretical and numerical studies, e.g. [3][5][6][16][17][18][19]. These studies confirmed the ability of an ANN to realize the temporal synchrony on the time scales of few milliseconds even with sharp synchronization on the time scale of single spikes. In general, these studies simulated a population of integrate-and-fire (IAF) neurons with adequate interconnectivity. Their discussions highlighted the major role of excitatory interconnections to achieve a certain degree of synchronous activity. Tsodyks et al presented a notable study in [1]. They considered the non-linear (frequency dependent) synapses for the generation of synchronous firing. Their results showed that the incorporation of nonlinear synapses in recurrent networks provide the basis for the emergence of short-time synchronous activity.

A stochastic pulsed-coupled neural network was presented in [20] showing synchronous and asynchronous behavior. A reduced stochastic dynamic model of an interconnected population of excitatory neurons with activity-dependent synaptic depression was proposed in [21], the discussion was focused on the bistability of voltage activities as up and down states. This is believed to be also related to the states of temporal synchrony within the neural ensemble. However, these studies and others did not consider the potential effects of *stochastic dynamic synapses* on synchronization of neural activity in ANN. It has been shown that deterministic representation of the neural actions does not *model the biological*

¹ In the referred study, the mathematically known cross-correlation was not meant explicitly, instead the general sense of temporal correlation was meant.

neural behavior realistically [22][23][24]. In addition, Kröger showed in [25] that probabilistic option in neuroscience offers advantages over the deterministic one.

Hence, in this paper, we test the ability of a network comprising IAF neurons and stochastic synapses to realize the concept of synchrony (the temporal correlations) between the signals of grouped neurons as states of synchronous activity. The goals are: a) to construct an ANN, that when driven by trains of spikes should be able to transform input signals combined with background synaptic activity (here introduced as synaptic noise) into correlated outputs and b) to show the ability to sustain such a level of synchrony over a considerable time course T_{sync} . The proposed ANN with Integrate-and-Fire (IAF) neurons is interconnected via a modified version of the Stochastic synaptic model presented in [22]. This architecture is arbitrarily chosen that both neuronal and synaptic representations are realizing the dynamic behavior. Moreover, the stochastic nature of the synaptic model is believed to be more plausible describing the nature of living tissue, in contrast to the deterministic model used in [1]. It represents, in an abstract way, an intrinsic source of noise within the system. A Hebbian-based Reinforcement-like learning algorithm is introduced as well.

2 Modified Stochastic Synaptic Model (MSSM)

Neurons are modeled as leaky-IAF neurons usually used in such type of simulations [1]. Each neuron is described by its voltage membrane potential V , that followed the following dynamics:

$$\tau_V \frac{dV}{dt} = -V + E_{\text{psp}}, \quad (1)$$

where τ_V is the membrane time constant, and E_{psp} is the total observed excitatory postsynaptic potential from all pre-synaptic terminals. Equations [1] and [5] are implemented as discrete forms introduced by [26] using Impulse Invariant Transform in order to facilitate the computations (Please refer to the articles [27][26] for the derivation).

Each synaptic connection is modeled as a stochastic activity-dependent connections using the synaptic stochastic model (SSM) [23]. This model estimates the transmission probability of an arriving action potential, i.e. spike, from a presynaptic neuron via a synapse to a postsynaptic neuron. The probability-of-release involved is governed by two counteracting mechanisms: facilitation and depression. Facilitation reflects the Ca^{2+} concentration in the presynaptic neuron, while depression represents the effect of the concentration of ready-to-release vesicles in the pre-synaptic neuron. The probability that the i th spike in the spike train triggers the release of a vesicle at time t_i at a given synapse is given by:

$$P(t_i) = 1 - e^{(-C_C(t_i) \cdot C_V(t_i))}, \quad (2)$$

where $C_C(t_i)$ and $C_V(t_i)$ represent the facilitation and depression mechanisms respectively at t_i . $C_C(t)$ and $C_V(t)$ are expressed mathematically as follow [24]:

$$C_C(t) = C_{Co} + \sum_{t_i} \alpha e^{-(t-t_i/\tau_{C_C})} \quad (3)$$

$$C_V(t) = \max(0, C_{Vo} - \sum_{t_i} e^{-(t-t_i/\tau_{V_C})}) \quad (4)$$

In eq. 3, τ_{C_C} and α represent the decay constant and the magnitude of the response respectively. C_{Co} represents the initial concentration of Ca^{2+} in the pre-synaptic terminal. In eq. 4, $C_V(t)$ is the expected number of vesicles of neurotransmitter molecules (Nt) in the ready-for-release pool at time t . C_{Vo} is the max. number of vesicles that can be stored in the pool. τ_{V_C} is the time constant for refilling the vesicles. A discrete-time version of those equations is used in simulation [23].

Recalling that the binding process of Nt on the postsynaptic membrane induce E_{psp} . Thus, E_{psp} can be expressed as follows [26]:

$$\tau_{epsp} \frac{dE_{psp}}{dt} = -E_{psp} + C_{Nt}, \quad (5)$$

where τ_{epsp} is a decay time-constant. C_{Nt} is the concentration of the Nt in the synaptic cleft. We assume that the latter can be estimated by tracing the amount of vesicles of Nt that remains in the presynaptic neuron, $C_V(t)$, over time. We introduce the following equation to estimate $C_{Nt}(t)$ and consequently couple the SSM with the IAF neuron model:

$$C_{Nt}(t) = \max(0, C_V(t - \Delta t) - C_V(t)) + C_{Nt}(t - \Delta t) e^{\frac{-\Delta t}{\tau_{Nt}}} \quad (6)$$

In eq. 6, C_{Nt} is the summation of: a) the estimated amount of Nt added with each release at any time step t (or the decrease in $C_V(t)$ over the time period Δt); where the $\max(\dots)$ avoids negatives and b) the amount of Nt that remains in the cleft from previous releases, namely at $t - \Delta t$. The decay with τ_{Nt} reflects the biological cleaning action, or the removal of the Nt from the cleft. By adding eq. 6 to SSM, we refer to it as the Modified SSM (MSSM). Using eq. 5 and 6, the IAF neuron model can be coupled to the SSM.

3 Network and Simulation

Up to our knowledge, the biologically accepted network size, in which temporal correlation can be observed and effectively utilized, is not precisely specified [1]. However, there are some hypothetical suggestions discussing the tenability of the network size. Singer for example analyzed the major factors affecting the ability of a group of neurons to exhibit synchronous activity [13]. He pointed out that the network size could be as small as two mutually coupled neurons and may be up to 100 neurons. Herzog and Gerstner argued that if synchrony is an essential feature for the brain activities, it should also be feasible in small networks [28]. They called this “the small network argument”, or the new benchmark for consciousness. Thus, they reported, there is a minimal model or a small network

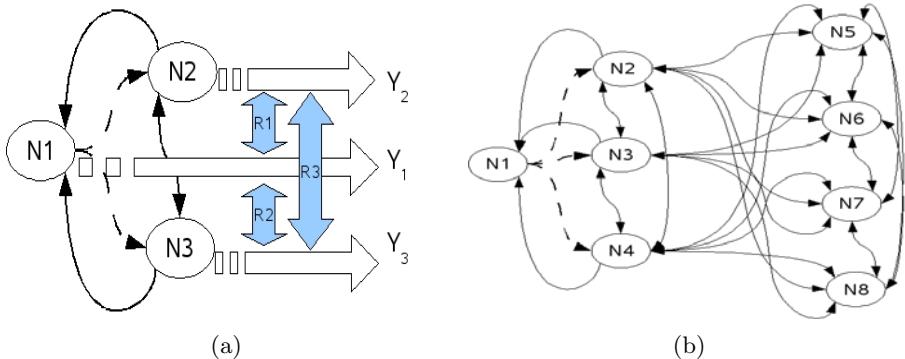


Fig. 1. Network schematic. a) 3 neuron network. The dashed lines are those synapses permitted to be trained. Double arrowed connections represent a mutual connection. Horizontal-right oriented arrows: output signals from each neuron: Y_1 , Y_2 and Y_3 . Vertical-two-headed arrows: the corresponding cross-correlation coeff.: R_1 , R_2 and R_3 . The trained synapses are from N_1 to N_2 and N_3 . b) 8 neuron network. The dashed lines are those synapses permitted to be trained. Double arrowed connections represent a mutual connection. The details of the outputs and the calculation of cross correlation are omitted for clarity. The trained synapses are the synapses from N_1 to N_2 , N_3 and N_4 .

that satisfies the criteria underlying consciousness, e.g. temporal synchrony, but is not conscious itself. They stated that groups of up to seven neurons are sufficient to realize memory, learning, or synchrony. Based on the analysis done in [29], a network of two neurons should be able to achieve spike-to-spike synchrony when enough mutual conductance is available.

Thus, two network structures are used in this study. A schematic of the first introduced network is in Fig. 1(a) with the input being fed only to the first neuron, N_1 . The network consists of 3 mutually interconnected neuron with only excitatory synapses. Also a bigger network is used with 8 neuron as in Fig. 1(b). In the latter one, the input is also fed to N_1 , and similar to the smaller network, the feed back to the input neuron is only possible from the neighboring neurons; in this case from N_2 , N_3 and N_4 . All synaptic connections are MSSM synapses as described in section 2 supported with white gaussian noise generators. The input is a set of 200 trains of spikes, each with a Poisson distributed inter-spike intervals for an epoch of 150 and 100 msec at 1 msec discretization for the 3 neuron and the 8 neurons networks respectively. This time epoch is arbitrarily used as a median value for the proposed time scale of T_{Sync} over which synchrony is plausible. The spike generator is adjusted to generate spikes with a maximum overall firing-rate of 300 Hz. Meanwhile, at each synapse a white Gaussian noise is added locally to the induced postsynaptic potential from this synapse. The level of the noise is modulated via simulated linear amplification.

For representing synchrony, the cross-correlation based measures are accepted in the detection of similarities in responses and for synchrony [119]. A correlation-based measure is introduced in [30] that calculates the cross-correlation coefficient

between neural responses after applying a Gaussian filtration on the responses. Here, the max. of cross-correlation coefficients between the filtered signals is used to indicate the degree of synchrony. The width of the Gaussian filter is chosen to be equal to the chosen neuronal refractory period of 2 msec.

4 Learning Rule

Generally, if no learning is implemented, the input signal and noise are fed to the network. At the end of each epoch, the mean R_m of the max. cross-correlation coefficients is calculated from all possible combinations between the responses from the three neurons. For example in the case of the network with 3 neuron $R_m = \text{mean}(R_1, R_2, R_3)$, where R_1 is the max. cross correlation coefficient between the Gaussian filtered versions of Y_1 and Y_2 . Similarly R_2 and R_3 , as in Fig. 1(a). An analog approach is used with the bigger network.

A Hebbian-based learning rule is introduced in [31,26] showing how both the timing parameters and constants can be updated based on the spiking activity of pre- and postsynaptic neurons. Here, this rule is extended to MSSM parameters. Specifically, the dynamics of synaptic or neural activities are governed through the contribution of electro-chemical mechanisms. Each of them is represented via a value, m , i.e. α in eq. 3 represents the max. allowed incurrent of Ca^{+2} ions to the presynaptic terminal [22]. A mechanism m could be either excitatory or inhibitory. According to the pre- and postsynaptic activity, the value of m is either increased or decreased following the Hebbian approach [31]. The update of the contribution values could be mathematically formed as follow: $m_{\text{new}} = (1 \pm r)m_{\text{current}}$, where r is the learning rate. In the proposed MSSM, such parameters are for example $\tau_{Nt}, \tau_{C_C}, \tau_{V_C}, \alpha, C_{Co}$ and C_{Vo} .

We introduce a feedback parameter, K , that represents the advance in the direction of getting both more and stable synchrony between the responses (i.e. a higher cross-correlation coefficient). Thus, it is the difference in the observed synchrony R_m from the current run and the previous one, mathematically expressed as follows:

$$K = R_m_{\text{current}} - R_m_{\text{previous}} \quad (7)$$

K is used as a modulator to the learning rate. Thus, the learning rule can be rewritten as follows:

$$m_{i\text{new}} = (1 \pm r \cdot K) m_{i\text{current}} \quad (8)$$

K can reverse the direction of the updating process of the parameters since it is a signed value, and can either accelerate or decelerate the learning process. This learning rule has the implicit objective of correlating the outputs corresponding to the same input *properties*. It emphasizes the sensitivity of the network to temporal and statistical properties embedded in input signals [26]. In this study, only forward MSSM synapses are allowed to update their parameters via this rule as illustrated in Fig. 1(a) and 1(b).

5 Results

Figures 2(a) and 2(b) illustrate the performance of the networks during the simulation with and without learning. In each figure, the two different traces can be taken as an indicator for two *neuronal states of synchrony*: Ground-State (Learning Off) and Active-State (Learning On). The network needed about from 20 - 40 learning runs until it reached the Active-State. The introduced networks are able, via the proposed modified stochastic synaptic model and the learning algorithm, to show two states of synchrony over a time window of 150 and 100

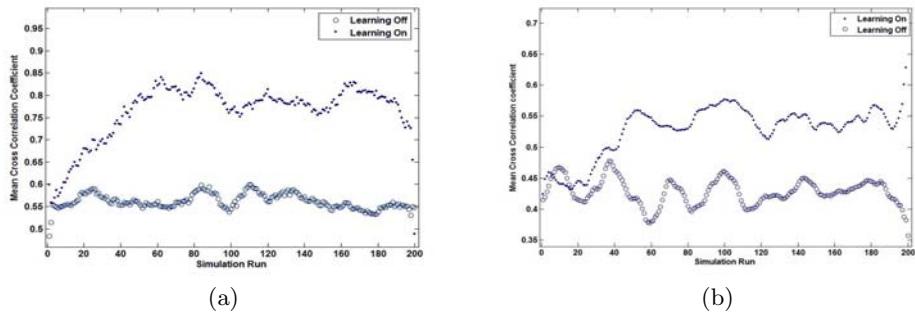


Fig. 2. Simulation result. a) The detected level of internal synchrony of the 3 neuron network in two cases: when learning is allowed, and when not allowed. $T_{sync} = 150$ msec. b) The detected level of internal synchrony of the 8 neuron network in two cases: when learning is allowed, and when not allowed. $T_{sync} = 100$ msec.

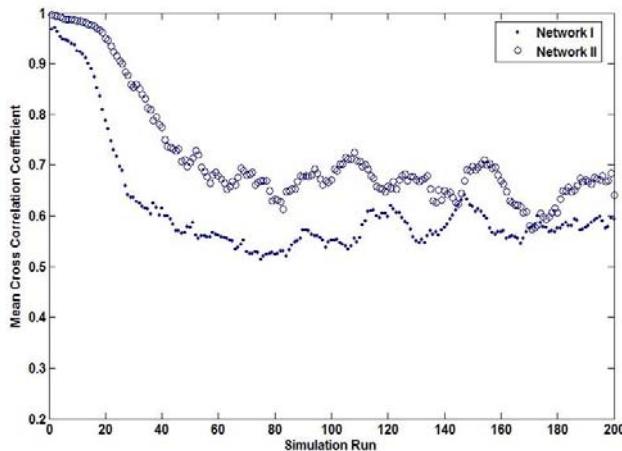


Fig. 3. Desynchronization. The networks reach a stable level of synchrony lower than the starting level. Results from both networks, 3 and 8 neuron (Network I and network II respectively), are illustrated.

msec in 3 neuron and 8 neuron network respectively. This agrees with the concept introduced in [12].

However, a network that achieves only synchrony is not so useful unless it can desynchronize its activity [15]. Hence, the ability of the network to desynchronize itself is also investigated. By setting the input to zero, the only remaining input is the noise which is equal for all synapses. Thus, the neurons start with typical firing patterns, this can be (obviously) seen in Fig. 3. The two traces of the cross-correlation coefficient start almost with a value of 1. Since learning is on, and while the network is trying to reach a general stable level of synchrony, the networks desynchronized itself reaching a middle level near those upper ones achieved as in Fig. 2(a) 2(b).

6 Discussion and Conclusion

The simulations presented here demonstrate that networks of neurons interconnected with stochastic synapses have a real tendency to realize special regimes of activity with synchronous discharge over biologically tenable periods of time. The simulation here is restricted to excitatory connections based on the mentioned discussion in the introduction section, however, the role of synaptic depression or specific inhibitory connections in case of using stochastic synapses needs further investigation.

Considering the network size in this study, it can not represent a cortical minicolumn [1] consisting of neurons, however, it could be viewed, in the context of synchronous activity, to have similar receptive field properties [14]. Other possibilities are issues of further study, e.g. it is expected that in larger simulated networks, such as a cortical mini- and hyper-column, where interconnections between neurons reflect their receptive field properties, other profiles of activity, may be with multiple internal states or with shorter time course, could exist.

As for the learning algorithm, and up to our knowledge, there may be no clear analytical evidence that biological neural systems can turn learning on and off. However, the logical analysis as in [8] still accepts that the biological neural circuitry can perform something similar to control, e.g. the control over the flow of information, the task sharing and non-physical rewiring of neural ensembles. This makes the idea of using the on/off learning rule acceptable.

In this paper, an ANN of IAF neurons coupled via MSSM synapses is introduced. In case of the evoked simulations, the network is driven by Poisson distributed trains of spikes and white-Gaussian noise. The latter is fed to synaptic activities. Considering that the Poisson distributed input represents a neural activity that carries certain information, the change in the level of synchrony could be seen as if the network, is likely to be, *memorizing* or *internally recalling* this input by pushing all its activities to *sync* with it. On the other hand, and in the case of spontaneous activity as the input is set to zero, the networks are able to find a lower stable state of synchrony, since they desynchronize their firing pattern. Both synchrony and desynchrony cope with the conceptual postulations discussed in the context of synchrony [12] and the role of noise in neural

information processing [5]. Hence, the proposed framework achieved successfully the general sense of sustaining a defined state of synchronous activity within a group of neurons over a considerable time course of 100 - 150 milliseconds.

It remains, however, that the expressive power of the proposed dynamics in terms of the number of achievable states is to be tackled. Besides, both the qualitative and quantitative sense of the difficulty to reach a certain generic state is an open questions as well. Moreover, the comparison between the abilities of the proposed dynamics versus the deterministic models to realize these states of synchrony remains to be discussed.

References

1. Tsodyks, M., Uziel, A., Markram, H.: Synchrony generation in recurrent networks with frequency-dependent synapses. *J. Neurosci.* 20, 50 (2000)
2. König, P., Engel, A.K., Roelfsema, P.R., Singer, W.: How precise is neuronal synchronization? *Neural Comput.* 7(3), 469–485 (1995)
3. Hansel, D., Mato, G., Meunier, C.: Synchrony in excitatory neural networks. *Neural Comput.* 7(2), 307–337 (1995)
4. Sejnowski, T.J., Paulsen, O.: Network Oscillations: Emerging Computational Principles. *J. Neurosci.* 26(6), 1673–1676 (2006)
5. Fellous, J.M., Rudolph, A.M., Destexhe, B.A., Sejnowski, T.J.: Synaptic background noise controls the input/output characteristics of single cells in an in vitro model of in vivo activity. *Neuroscience* 122, 811–829 (2003)
6. Lindner, B., Schimansky-Geier, L., Longtin, A.: Maximizing spike train coherence or incoherence in the leaky integrate-and-fire model. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 66(3 pt. 1), 031916 (2002)
7. Kreuz, T., Haas, J., Morelli, A., Abarbanel, H.D., Politi, A.: Measuring spike train synchrony and reliability. *BMC Neuroscience* 8(suppl. 2), 79 (2007)
8. Durstewitz, D.: Self-Organizing Neural Integrator Predicts Interval Times through Climbing Activity. *J. Neurosci.* 4(12) (2003)
9. Singer, W.: Neuronal synchrony: a versatile code for the definition of relations. *Neuron* 24, 49–65 (1999)
10. Gray, C.M.: The temporal correlation hypothesis of visual feature integration: Still alive and well. *Neuron* 24(1), 31–47 (1999)
11. Salinas, E., Sejnowski, T.J.: Correlated neuronal activity and the flow of neural information. *Nature Reviews Neuroscience* 2, 539–550 (2001)
12. von der Malsburg, C.: The what and why of binding: The modeler's perspective (1999)
13. Singer, W., Gray, C.M.: Visual feature integration and the temporal correlation hypothesis. *Annu. Rev. Neurosci.* 18, 555–586 (1995)
14. Averbeck, B.B., Latham, P.E., Pouget, A.: Neural correlations, population coding and computation. *Nature Reviews Neuroscience* 7, 358–366 (2006)
15. Campbell, S.R., Wang, D.L., Jayaprakash, C.: Synchrony and desynchrony in integrate-and-fire oscillators. *Neural Computation* 11(7), 1595–1619 (1999)
16. Neltner, L., Hansel, D.: On synchrony of weakly coupled neurons at low firing rate. *Neural Comput.* 13(4), 765–774 (2001)
17. Bressloff, P.C., Coombes, P.S.: Dynamics of strongly coupled spiking neurons. *Neural Comput.* 12(1), 91–129 (2000)

18. Vreeswijk, C.V., Hansel, D.: Patterns of synchrony in neural networks with spike adaptation. *Neural Comput.* 13(5), 959–992 (2001)
19. Mikula, S., Niebur, E.: Rate and synchrony in feedforward networks of coincidence detectors: Analytical solution. *Neural Comput.* 17(4), 881–902 (2005)
20. DeVille, R.E.L., Peskin, C.S.: Synchrony and Asynchrony in a Fully Stochastic Neural Network. *Bulletin of Mathematical Biology* 70(6), 1608–1633 (2008)
21. Holcman, D., Tsodyks, M.: The Emergence of Up and Down States in Cortical Networks. *Science* 2(3), 174–181 (2006)
22. Maass, W., Zador, A.M.: Dynamic stochastic synapses as computational units. *Neural Computation* 11, 903–917 (1999)
23. Natschläger, T.: Efficient Computation in Networks of Spiking Neurons Simulations and Theory. PhD thesis, Institute of Theoretical Computer Science, Austria (1999)
24. Natschläger, T., Maass, W., Zador, A.: Efficient temporal processing with biologically realistic dynamic synapses. *Computation in neural system* 12, 75–78 (2001)
25. Kroger, H.: Why are probabilistic laws governing quantum mechanics and neurobiology? *Solitons and Fractals* 25, 815 (2005)
26. Namarvar, H.H., Liaw, J.S., Berger, T.W.: A new dynamic synapse neural network for speech recognition. *IEEE Trans.*, 2985–2990 (2001)
27. Liaw, J.S., Berger, T.W.: Dynamic synapse: A new concept of neural representation and computation. *Hippocampus* 6, 591–600 (1996)
28. Herzog, M.H., Esfeld, M., Gerstner, W.: Consciousness & the small network argument. *Neural Networks* 20(9), 1054 (2007); *Brain and Consciousness*
29. Dodla, R., Wilson, C.: Synchrony-asynchrony transitions in neuronal networks. *BMC Neuroscience* 9(suppl. 1), 9 (2008)
30. Schreiber, S., Fellous, J.M., Whitmer, D., Tiesinga, P., Sejnowski, T.J.: A new correlation-based measure of spike timing reliability. *Neurocomputing* (52-54), 925–931 (2003)
31. Liaw, J.S., Berger, T.W.: Computing with dynamic synapses: A case study of speech recognition. In: Proc. IEEE Int. Conf. Neural Networks, pp. 352–355 (1997)

Coexistence of Cell Assemblies and STDP

Florian Hauser, David Bouchain, and Günther Palm

Institute of Neural Information Processing, Ulm University, Germany

florian.hauser@uni-ulm.de

Abstract. We implement a model of leaky-integrate-and fire neurons with conductance-based synapses. Neurons are structurally coupled in terms of an ideal cell assembly. Synaptic changes occur through parameterized spike timing-dependent plasticity rules which allows us to investigate the question whether cell assemblies can survive or even be strengthened by such common learning rules. It turns out that for different delays there are parameter settings which support cell assembly structures and others which do not.

1 Introduction

One of the most fundamental features of the brain is its ability to change over time depending on sensation and feedback, i.e. its ability to learn, and it is widely accepted today that learning is a manifestation of the change of the brain's synaptic weights according to certain rules. In 1949, Donald Hebb postulated that repeatedly correlated activity between two neurons enhances their connection ([1]), leading to what is today called Hebbian cell assemblies—a strongly interconnected set of excitatory neurons. These cell assemblies (CA) can be used to model working memory ([2]) in the form of neural auto-associative memory and thus may provide insight into how the brain stores and processes information ([3]). Cortex modeling based on cell assemblies has been investigated theoretically as well as applied to robot control in e.g. [4].

That synaptic weights do indeed change with correlated pre- and post-synaptic activity has been discovered in the seventies (e.g. [5]). Recent research however indicates that the temporal order of pre- and post-synaptic spikes is also important ([6]), i.e. that pre-synaptic before post-synaptic activity leads to potentiation (an increase in synaptic efficacy), whereas the reverse case causes depression of synaptic weights. This phenomenon has been termed spike timing-dependent plasticity (STDP) ([7]) and has been investigated numerous times physiologically as well as computationally (see for example [8,9,10]).

Here, we examine Hebbian cell assemblies and their stability with respect to STDP. While the effect of the STDP rule on local feed forward structures, e.g. Abelesian synfire chains ([11]) is relatively easy to understand, it is not immediately obvious how highly recurrent structures like cell assemblies are affected. In circuits with direct feedback, like CAs, a neuron's spike falls into both the positive and negative part of the STDP window and thus, even if the neuron receives input directly from itself and therefore takes part in the production of

its own spikes, it is not intuitively clear whether its synapses are strengthened or weakened.

The primary question we strive to answer is: Under what computational circumstances, i.e. which parameter settings, can cell assemblies coexist with the STDP learning rule?

In the following, we investigate how an “ideal” cell assembly, i.e. a set of model neurons interconnected through a weight matrix C , $c_{ij} = c \forall i, j$ with a delay matrix D , $d_{ij} = d \forall i, j$ is affected by STDP. We show that whether or not cell assemblies can coexist with the STDP rule depends on the type of STDP window used, the initial weight matrix as well as the delay matrix.

2 Model

We consider a network of N leaky-integrate-and-fire neurons. These are completely interconnected through conductance-based synapses. For $j = 1, \dots, N$ the subthreshold dynamics of neuron j is given by

$$\tau_m \dot{v}_j = v_{\text{rest}} - v_j + g_j \cdot (E - v_j) + \mathcal{N}(0, 1) \quad (1)$$

as proposed in [7]. v_j describes the membrane potential of neuron j , v_{rest} its resting potential, g_j the corresponding conductance of the neuron’s excitatory inputs, and E names the reverse potential. \mathcal{N} denotes Gaussian noise added to the membrane dynamic. If a membrane potential reaches a certain threshold v_{th} , the point in time is stored and the potential is reset to a global potential v_{reset} and g_j is set to 0. The network’s delay D is the same for all neurons according to the definition of a cell assembly. We assume an existing cell assembly with synapses which are strong enough to generate self-sustained activity. Synaptic plasticity occurs through STDP, implemented through traces of pre- and post-synaptic activity. Four parameters are necessary to define a proper STDP time window: τ_+ , τ_- , A_+ and A_- . These specify time constants and maximums of change to synaptic strength. We use the additive STDP-update-rule as physiological tradeoff ([12]). Traces are modeled by two simple ODEs of the form

$$\tau_+ \dot{P}_{ij} = -P_{ij} \quad (2)$$

and

$$\tau_- \dot{M}_j = -M_j \quad (3)$$

where P_{ij} characterizes the pre-synaptic trace from neuron i to j and M_j its post-synaptic counterpart. Synaptic conductances also follow this pattern by

$$\tau_g \dot{g}_j = -g_j . \quad (4)$$

The $N \times N$ matrix C stores synaptic strengths. If a pre-synaptic spike occurs, P_{ij} is increased by an amount A_+ and C_{ij} by $M_j \cdot g_{\max}$ where g_{\max} refers to the maximum possible conductance. The delay D must be included in the calculation. After updating the post-synaptic trace M_j by A_- for every incoming

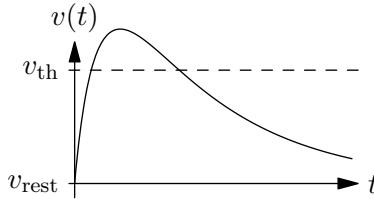


Fig. 1. The membrane potential curve (“EPSP shape”) as given by Eq. (1)

spike, C_{ij} will be modified by adding $P_{ij} \cdot g_{\max}$. So while no spike arrives, all traces just decay exponentially, whereas otherwise the traces will be updated by an amount and added on synaptic strengths which again influence the change of the total conductance g_j at neuron j . The model parameters and their respective values are given in Table 1. The solution to Eq. (1), i.e. an EPSP at time $t = 0$ is shown in Fig. 1.

Table 1. Constant parameters used for the network model

Parameter Value Description	
N	300 Number of neurons
τ_m	0.02 Membrane time constant
τ_g	0.005 Conductance decay time constant
g_{\max}	0.015 Maximum conductance per synapse
v_{rest}	-70 Resting potential
v_{reset}	-80 Reset potential
v_{th}	-54 Threshold potential
E	30 Reverse potential

A_+ , A_- , τ_+ , τ_- and D will be specified in Section 4 because they span the parameter space. There are also two cases for initial synaptic strengths which will be described later.

3 The Integration Interval

In order to determine the parameter ranges for stability of cell assemblies analytically, we have to determine the integration interval, i.e. the time it takes from the EPSP onset to the spike generation threshold. To this end we simplify the model equation by setting v_{rest} to 0 and the threshold to $v'_{\text{th}} = v_{\text{th}} - v_{\text{rest}} = 16$. Furthermore, we substitute the term $g(E - v)$ with $100 \cdot g$. The membrane equation is then given by

$$\tau_- \dot{v}(t) = -v(t) + 100 \cdot g(t) , \quad (5)$$

with $g(t)$ of the form

$$g(t) = c \cdot e^{\frac{-t}{\tau_g}} \quad (6)$$

where c is the current synaptic strength (equal for all synapses in the network) multiplied by 300 (the number of incoming synaptic connections per neuron).

We can now relate the (approximate) time the membrane needs to reach the threshold to the synaptic connection strength c , given the synchronous arrival of all EPSPs at time $t = 0$, as well as the initial condition $v(0) = 0$. This is a function of t corresponding to the convolution of v with g

$$v(t) = 100 \cdot c \cdot \int_0^t e^{\frac{-\tau}{\tau_m}} \cdot e^{\frac{-t+\tau}{\tau_g}} d\tau \quad (7)$$

which shows that the EPSP amplitude is proportional to c . This means that the larger c is, the shorter the interval between EPSP onset and spike generation becomes.

Therefore, with a higher synaptic efficacy, the potentiation interval has a higher chance of outweighing the depression interval D . We show in Section 4 that this is indeed the case for our model.

4 Simulations

Our model in Section 2 was implemented in MATLAB, using parameter settings as given in Table 1. All ODEs were integrated by an explicit Euler method with fixed step size $h = 0.001$. Simulation time was $T = 10$ for each parameter triple (k, l, D) . We assume the integral over a fixed STDP window to be negative in order for long-term depression to outweigh LTP. So all parameter settings used in our simulations satisfy this constraint by choosing $\frac{1}{l} < k$, where $k = \frac{A_-}{A_+}$ and $l = \frac{\tau_-}{\tau_+}$. This can be directly derived from

$$-\int_{-\infty}^{\infty} A_- e^{\frac{t}{\tau_-}} dt + \int_{-\infty}^{\infty} A_+ e^{-\frac{t}{\tau_+}} dt < 0 \quad (8)$$

where k and l name the parameters which are used to visualize our results. So, k takes values in the range -1.05 to -0.25 with step size 0.4 , l between 1.0 and 5.0 with step size 1.0 . Hence, simulation starts with a configuration $(-1.05, 1, \cdot)$ corresponding to the STPD time-window used in [7] and ends with estimated values $(-0.25, 5, \cdot)$ obtained through physiological experiments ([13]). Because k and l are defined in each case as ratios of time constants and amplitudes, one part of the fraction has to be fixed. We used $\tau_+ = 0.02$ and $A_+ = 0.005$ and varied k and l respectively. Additionally this was done with varying delays D in the range 1.0 to 30.0 with constant step size 5.0 which gives us a three-dimensional parameter space.

Moreover, there are two initial synaptic strength settings. For a just new formed CA, we used uniform strengths of size 0.005 . On the other hand, strengths were set to $g_{max} = 0.015$ for an already longer existing CA. On the start of an simulation, all neurons were forced to emit one spike synchronously to activate the CA. After that, membrane and synaptic dynamics determine the chronological sequence by itself.

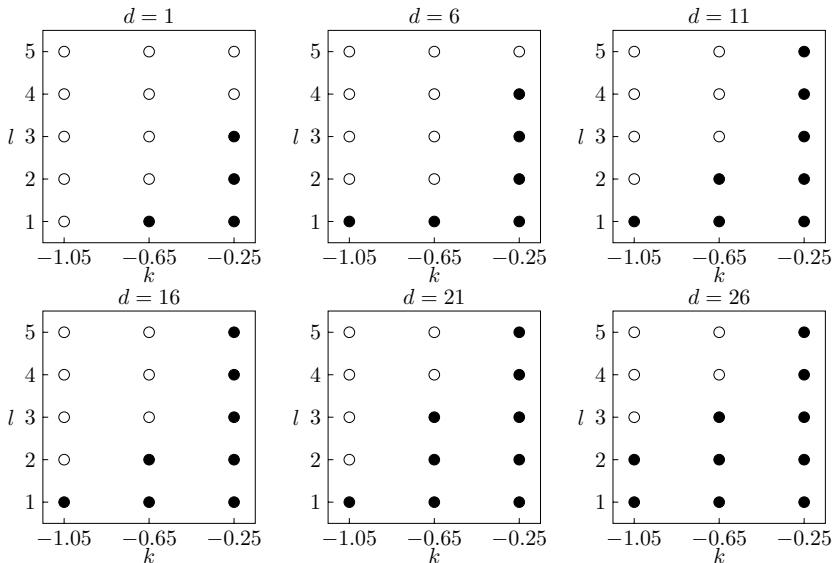


Fig. 2. Simulation results for strong initial synaptic efficacy. Circles denote a setting for which the CA activity dies, dots indicate parameters which lead to stabilization. The larger the network delay, the higher the chance of potentiation to outweigh depression.

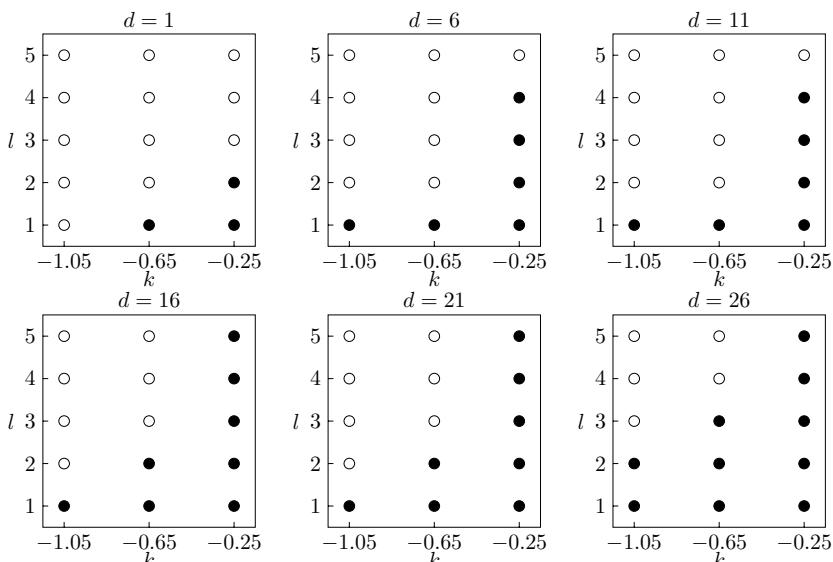


Fig. 3. Same as Figure 2, but weak initial synaptic efficacy, i.e. the coupling is just strong enough to evoke postsynaptic spikes

To decide if a CA still exists after simulation time, some kind of stability criterion is needed. This can be defined by looking at the time interval $[T - D, T]$ and counting all spikes which therein occur. If this number of spikes matches N , the CA still exists.

Thus, we get a series of plots for different delays with initial strong synaptic strengths in Figure 2 and one for weak ones in Figure 3. Dots correspond to a stable CA whereas circles indicate a parameter setting where the activity dies. As we can observe, there are parameter settings for each case where a CA does exist or not.

5 Discussion and Conclusions

Due to the highly recurrent network topology, each neuron's spikes contribute to long-term potentiation and long-term depression. Whether LTP outweighs LTD, or vice versa, depends on a number of model parameters. In a symmetrical STDP window, i.e. when $A_+ = A_-$ and $\tau_+ = \tau_-$, this depends essentially on the time interval between spikes, which is the sum of the potentiation interval and the network delay. The potentiation interval corresponds to the difference in time between spike generation and EPSP onset, which we will henceforth refer to as Δ_e . Depression is based on the reverse, i.e. the difference between EPSP onset and spike generation (of the previous spike), which is exactly the network delay D . Thus, if a spike evocation on EPSP arrival takes longer than the time the same spike needs to arrive back at the neuron, the synapse is weakened and the activity in the cell assembly would ultimately diminish.

Taking into account the possibility of asymmetry and a negative integral of the STDP window leads to different behavior. With the window parameters used in [7], for example, Δ_e has to be smaller than D for all D if CA structures are to remain stable, since the STDP function will always be larger for $-\Delta_e$ than it is for Δ_e . If the parameters are chosen similar to those presented in [13], i.e. such that the peak amplitude for LTP is far greater than for LTD and the negative integral constrained is adhered to by making the time constant τ_- much larger than τ_+ , then D can be shorter than Δ_e .

Although our experiments include just leaky integrate-and-fire neurons and a simple model of synaptic dynamics, we can observe important factors which influence the existence of Hebbian CA networks in the context of STDP. Our implementation leads to what can be called a “population EPSP”, i.e. a single EPSP scaled by the number of incoming synapses. More biologically plausible modeling could be achieved by additional changes to parameters like C , D , initial membrane potential values or noise added to the system. We obtained well defined limits of parameter settings for existence as well as non-existence of CAs.

It can be argued that the existence of CAs in the brain is very unlikely because even small parameter changes to the STDP window may lead to disintegration of the CA. Thus, there must not occur to much unspecific (i.e. uncorrelated with the assembly activity) synaptic input during formation of CAs, as otherwise destabilization would occur instantly.

CAs are often assumed to be a very local structure consisting of strongly interconnected excitatory neurons. The local circuit is spatially small, and thus has small delays. However, our results suggest that CAs are more likely to form with a longer delay, and are more likely to disappear with shorter delays.

On the other hand, the fact that longer delays are better for the stability of cell assemblies under the STDP regime suggests that CAs are more likely to be global, i.e. long-range cortico-cortical structures. Another possibility is that, depending on factors like initialization of the membrane potentials or a randomly generated delay matrix, not all neurons of a CA might fire at the same time, which leads to a kind of synfire loop activation moving across the assembly, thereby avoiding the phenomenon that the same spike falls into both the potentiation and the depression window of STDP.

References

1. Hebb, D.O.: *The Organization of Behavior*. Wiley, Chichester (1949)
2. Durstewitz, D., Seamans, J.K., Sejnowski, T.J.: Neurocomputational models of working memory. *Nat. Neurosc.* 3, 1184–1191 (2000)
3. Palm, G.: *Neural Assemblies: An Alternative Approach to Artificial Intelligence*. Springer, Heidelberg (1982)
4. Markert, H.: *Neural Associative Memories for Language Understanding and Robot Control*. PhD thesis, Ulm University (2008)
5. Lmo, T.: The discovery of long-term potentiation. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 358, 617–620 (2003)
6. Bi, Q.Q., Poo, M.M.: Synaptic modification in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosc.* 18, 10464–10474 (1998)
7. Song, S., Miller, K.D., Abbott, L.F.: Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosc.* 3(9), 919–926 (2000)
8. Rossum, M.C.W., Bi, Q.Q., Turrigiano, G.G.: Stable Hebbian learning from spike timing-dependent plasticity. *J. Neurosc.* 20, 8812–8821 (2000)
9. Burkitt, A.N., Gilson, M., van Hemmen, J.L.: Spike-timing-dependent plasticity for neurons with recurrent connections. *Biol. Cyb.* 96, 533–546 (2007)
10. Hosaka, R., Araki, O., Ikeguchi, T.: STDP provides the substrate for igniting synfire chains by spatiotemporal input patterns. *Neur. Comp.* 20, 415–435 (2008)
11. Abeles, M.: *Corticonics: Neural Circuits of the Cerebral Cortex*. Cambridge University Press, New York (1991)
12. Morrison, A., Diesmann, M., Gerstner, W.: Phenomenological models of synaptic plasticity based on spike timing. *Biol. Cyb.* 98, 459–478 (2008)
13. Dan, Y., Poo, M.M.: Spike timing-dependent plasticity: From synapse to perception. *Physiol. Rev.* 86, 1033–1048 (2006)

Controlled and Automatic Processing in Animals and Machines with Application to Autonomous Vehicle Control

Kevin Gurney¹, Amir Hussain², Jon Chambers¹, and Rudwan Abdullah²

¹ Adaptive behaviour research group, Department of Psychology,
University of Sheffield, S10 2TN, UK
k.gurney@shef.ac.uk

² Department of Computing Science, University of Stirling, Stirling,
FK9 4LA, Scotland, UK

Abstract. There are two modes of control recognised in the cognitive psychological literature. *Controlled* processing is slow, requires serial attention to sub-tasks, and requires effortful memory retrieval and decision making. In contrast *automatic* control is less effortful, less prone to interference from simultaneous tasks, and is driven largely by the current stimulus. Neurobiological analogues of these are *goal-directed* and *habit-based* behaviour respectively. Here, we suggest how these control modes might be deployed in an engineering solution to Autonomous Vehicle Control. We present pilot data on a first step towards instantiating automated control in the architecture, and suggest a synergy between the engineering and biological investigation of this dual-process approach.

Keywords: Executive control, habits, basal ganglia loops, Fuzzy Tuning, Autonomous Vehicle Control, dual-process theory.

1 Introduction

What are the different control strategies used by humans to enact behaviour and what might their implications be for control engineering? To get some idea of an answer to the first question, consider the following scenarios. Imagine making tea soon after getting out of bed in the morning in your own kitchen. You probably know exactly what to do without having to consciously be aware of it – the location of the tea, milk, sugar, kettle and water-tap are all well learned, as is the motor actions required to interact with the objects in these locations. Introspection after the event leads us to use terms such as; ‘I did it in my sleep’ or ‘I was on auto-pilot’. Now consider doing the same task if you are staying at a friend’s house for the first time. A completely different strategy appears to be used. Thus, we have to be alert, explore, and use high level cognitive knowledge that we hope generalises well (for example, we hope the tea will be in a cupboard near the sink, not in the living room). These two modes of control are well recognised in the psychological literature [see, for example [1](#)] as *automatic*, and *controlled* or *executive* processing respectively. There is also

growing neurobiological evidence for the existence of different control regimes, supported by different neural substrates [2, 3].

In this paper, we first review the two modes of control from a biological perspective, looking at their characterisation, mechanistic and neural substrate, and the rationale for there being a control duality in the first place. We then go on to deploy these ideas in the context of an architecture for autonomous vehicle control which uses multiple sub-controllers [4]. This builds on a general consideration of the links between biology and the vehicle controller dealt with previously [5]). Thus, we suggest how the architecture may incorporate ‘automatised’ processing to complement its current ‘executive’ control of sub-controller selection. The programme of work required to fully evaluate the resulting architecture is ambitious, but we make a first step here by describing a pilot study dealing with one aspect of learning automatic responses. We conclude by considering the computational problems that arise in the new architecture and their similarity with biological counterparts.

2 Dual Mode Control: Biological Perspectives

2.1 Characterisation

The concepts of automatic and controlled processing have a long history in psychological research [6] but the development of a full *dual-process* theory of cognition is often attributed to the work of Shiffren and Schneider [7, 8]. Controlled processing is under the subjects direct and active control, is slow, and requires serial attention to component stimuli or sub-tasks. It is sensitive to a task’s difficulty (which limits the ability to perform additional tasks at the same time) and requires effortful memory retrieval and decision making. In contrast [9] automatic control is less effortful, less prone to interference from simultaneous tasks, is driven largely by the current stimulus and does not necessarily give rise to conscious awareness. As well as defining two kinds of processing, the dual-process theory supposes a dynamic of the transfer of control under learning. Thus, controlled processing is the mode required in the early acquisition of new skills which are, when well-practiced, carried out using automatic processing. For a recent review of dual-process theory see [10].

The development of automatic processing has close similarities with the notion of *stimulus-response* (S-R) learning, or *habit* learning in the behavioural neuroscience literature (for a recent review see [11]). A habit is deemed to be constituted when an animal responds to a stimulus without regard to the value of reward obtained by that response [12]. This may occur, for example, if the response elicits a food reward but the animal is sated. Thus, habits are context-triggered, learned behaviors which are entirely contingent on stimulus and not the goal. Similarly, controlled processing may be likened to *goal directed* behaviour in animals in which the animal makes a response eliciting something of genuine, current reward value. Further, just as there is a gradual transfer from controlled to automatic processing in cognitive dual-process theory, habits are learned after an initial goal-directed period of the same behaviour.

2.2 Mechanisms and Models: Psychological

In the domain of cognitive psychology, one of the most influential models of dual process theory is that of Norman and Shallice [13, 14] - henceforth referred to as the NS model. Central to the model (Figure 1) is the idea that behaviour is decomposed into a set of primitive actions or *schemas*. Examples of schemas

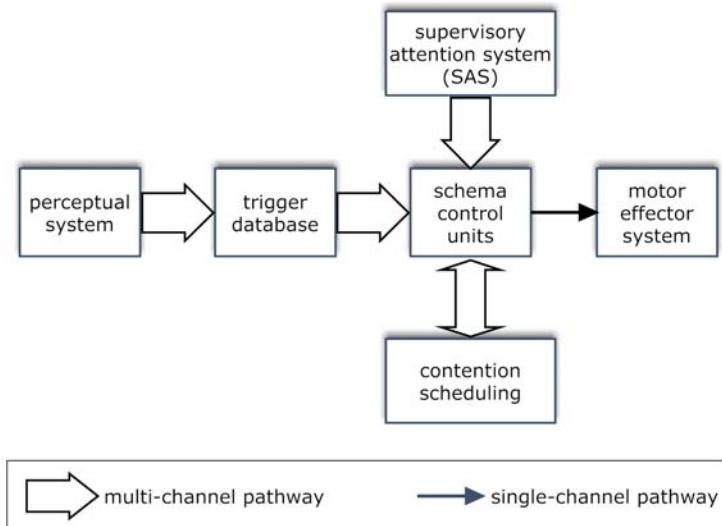


Fig. 1. The model of Norman and Shallice (1980) with Supervisory Attentional System (SAS)

are provided by the scenario described in the introduction: pouring milk into a cup, opening a fridge door, putting a kettle on, etc. These schemas are activated by sensory input (pre-processed through the perceptual system) via a set of 'triggers'. Thus, each schema delivers a particular behavioural output in response to a narrow range of sensory input.

However, given a particular environmental situation, several possible schemas may be activated. These can not all be expressed simultaneously because there are limited motor resources (one can't pour milk and spoon sugar with your dominant hand simultaneously). This problem of schema selection is an example of the more widely studied problem of *action selection* [15] - a universal problem for all autonomous agents. It appears in many forms across various disciplines, and the need for its resolution in all animals lies at the heart of our neuroscientific account of automatic and controlled processing described in the next section.

In the current context, schema selection is mediated in the first instance by a process called 'contention scheduling' which is supposed to occur via mutual inhibition between schema activation levels. In this way, schemas which are triggered strongly by sensory input are more likely to be enacted than those which are

weakly activated. However, what happens if the system doesn't have a schema triggered by the current sensory context, or the initially chosen schema fails to achieve the goal? Such non-automatic processing requires control by a more sophisticated, general purpose planning and programming system. In the NS model this is the so-called Supervisory Attention System (SAS). However, the SAS does not take over the role of the contention scheduling mechanism, rather, it works by *biasing* the results of that process on the schema pool.

The NS model has recently been reinvigorated in quantitative form by Garforth et al [16]. Here, key components of the NS scheme have been incorporated into a simulated autonomous robot model that 'forages for food'. The model is a connectionist one using clusters of neural networks to implement the various functional components which include but also extend the NS model framework.

2.3 Mechanisms and Models: Neurobiological

The Norman and Shallice model highlights the importance of action selection as a pivotal concept in understanding the functions of, and relationship between, controlled and automatic processing. In the vertebrate brain, action selection (cf contention scheduling) is, believed to be mediated by a set of sub-cortical structures known as the basal ganglia (BG). In [17], we developed this idea and argued that the BG act as a central switch, receiving requests for behavioural expression from subsystems throughout the brain, and then selectively permitting these to take control of their target motor plant (or cognitive resources).

The strength of an action request is supposed to reside in its overall signal level or 'salience' (cf schemata activity level in the NS model). Further, requests are more or less 'listened to' by the BG according to how well activity profiles on the afferents to BG input neurons match with the corresponding patterns of synaptic 'weights' [18]. Thus, a template match between the two will result in a BG action channel which is sensitive to the action request (this would correspond to assigning sensitivities to elements in the trigger database of the NS model). Further downstream, output from the BG is inhibitory, normally active, and targets the requesting subsystem in a closed loop (via thalamus). Actions are selected when competitive processes between action channels in the BG result in withdrawal of inhibition from the target subsystem. An action request will therefore be successful if its salience is sufficiently high, and there is a receptive action channel input in the BG. These processes are illustrated for two action channels in Fig. 2. The scheme shown in Fig. 2 is well suited to perform action selection for habitual or routine behaviours if the action requests are dominated by signals from sensorimotor cortical areas, representing environmental state and ongoing motor actions. Indeed, the role of the BG in the encoding of habits under these circumstances is well documented [11, 19]. In particular, we propose that habitual selection by pattern matching in BG works in a similar way to schemata triggering in the NS model.

But, if the basal ganglia is the neural substrate of contention scheduling in the NS model, what instantiates the SAS? Much evidence points to the pre-frontal-cortex (PFC) as serving an 'executive' or supervisory role similar to that

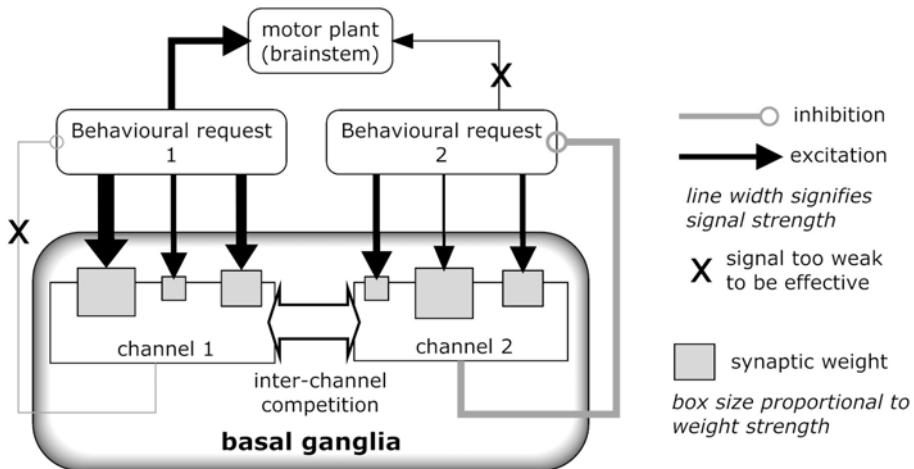


Fig. 2. The basal ganglia as the central action selection ‘switch’ in the brain, showing pattern matching at the input and closed loop architecture (thalamus omitted for simplicity). Channel 1 is selected because, i) its overall input salience is greater, and ii) its pattern of activity is better matched with the synaptic weights on the pertinent BG input neuron.

of the SAS [20]. Further, PFC also forms loops through BG (but with different domains of BG than sensorimotor cortices), and there is the possibility of substantial crosstalk from the PFC loops to their sensorimotor counterparts [21, 22]. However, just as in the NS scheme, the ‘supervisory’ PFC does not simply usurp the automatic processing system, but rather, works by modulating it via its cross-connections. In this way, it is supposed that goal-directed (non-habitual) behaviours governed by PFC can transfer into habits in sensorimotor loops by learning therein under the influence of the PFC loops [11, 19].

2.4 Automatic and Controlled Processing – The Computational Rationale

So far we have characterised the processes of controlled and automatic processing, and offered possible mechanistic and neurobiological accounts. However, it remains to answer the question: what computational advantages might ensue from having two modes of action selection? This is especially pertinent if we are to adopt a dual process model in a conventional engineering situation for, while developing biologically grounded solutions is interesting for its own sake, this is not reason enough to do so.

One obvious advantage to having routine, automatised control is that it offers a speed advantage. Thus, while all decisions could, in principle, be made using a high level supervisory system, if this were the case, every behavioural decision would require extensive high-level cognitive processing, which would incur a time

penalty by dint of the excessive computational load. In contrast, the automatised version of the behaviour is running with little computational overhead (using, say, pattern matching for example). Such decision time differences are observed extensively in cognitive tasks which tap both processes [23].

A second, less obvious advantage has been recently demonstrated by Daw et al [24]. They modelled learning a task using two training schemes. One made use of internal cognitive models and the other was based on immediately available information; these schemes are representative of controlled and automatic processing respectively. The result was that each method showed different *uncertainties* in the expected reward of each trial. Further, these uncertainties changed as learning progressed, and as a function of the time into the trial. Thus, an optimal agent could make use of different strategies to dynamically reduce uncertainty to a minimum.

3 Autonomous Vehicle Control

The field of autonomous vehicles is a rapidly growing one which promises improved performance, fuel economy, emission levels, comfort and safety. An important component of autonomous vehicle control (AVC) aims to control the throttle, wheel brake and steering systems so that the vehicle can follow a desired path and target speed (possibly governed by a ‘lead vehicle’) and at the same time keep a safe inter-vehicle spacing under the constraint of comfortable driving [25]. This is the problem considered here, and one particularly promising strategy for its control is to break the task space into a series of distinct operating regions and to switch between them when required [26]. In order to make contact with the previous discussion, we then interpret this control scheme as one in which the plant (an automobile) switches between distinct ‘behaviours’. Recently, Abdulah et al [4] have formulated one instantiation of AVC using such a multi-controller scheme. We have previously described similarities between this AVC architecture and biological solutions to action selection [5], but here, we focus on the issues of automatic and controlled behaviour.

As well as using multiple, sub-controllers, the AVC controller also uses a fuzzy logic based switching and tuning supervisor which uses information about the systems internal state, and the environment, to determine which controller to choose. It is this part of the controller which is relevant to the present discussion and it is shown in Figure 3a. Here, n controllers are shown for generality, although only 2 were used in the original scheme. Each controller is MIMO with three outputs – one for each of ‘steer’, ‘throttle’ and ‘brake’ – and the outputs of the i th controller form a vector $y^{(i)}$. In general, the selected signal for each control function may be independently chosen from any of the controllers (so ‘steer’ and ‘brake’, for example, may originate from different controllers). The result is a final control signal vector \mathbf{y}^* . The selection of the control signals $\mathbf{y}^*(t)$ is dependent on those at the previous time step $\mathbf{y}^*(t-1)$; nevertheless, the selection function *per se* is performed entirely by the fuzzy controller which, because of its powerful processing capacity, is a clear example of a supervisory executive

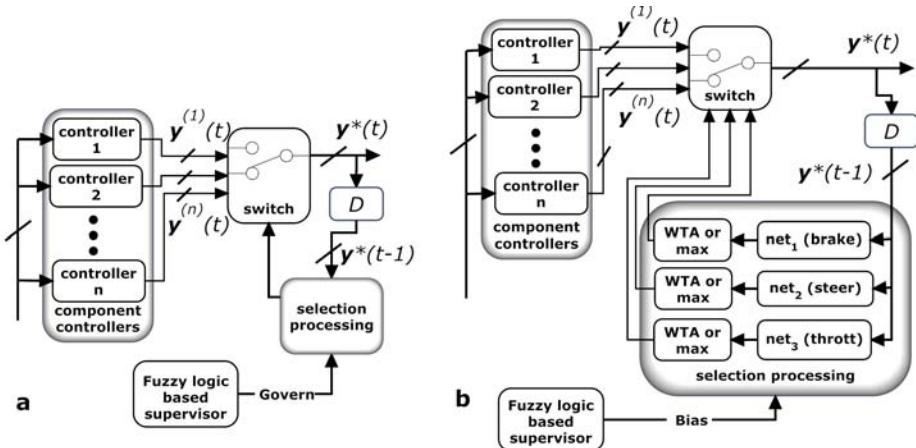


Fig. 3. Partial view of AVC architecture showing controller selection. a) original scheme, b) new scheme with automatised, or ‘habit-based’ control option.

system exercising controlled processing. There is, however, no counterpart in the AVC architecture to automatic, ‘habit-based’ processing, which might incur the benefits described in section 2.4).

In order to remedy this, we propose the architectural modification shown in Figure 3b, which is based on the discussion of the basal ganglia and its use of weight-input template matching to trigger action selection. That is, for each of the three control signals, a neural network is assigned which has, as its input, the signal vector $\mathbf{y}^*(t - 1)$. It is interesting to note that the use of loop with delay is reminiscent of the loops through basal ganglia (Figure 2). Each network has a number of output units z_j , $1 \leq j \leq n$, and has to learn to flag which controller should be used to drive each of the three control signals using a 1-out-of- n code (one unit on, the others off). In general, with many controllers, it will be necessary to subject the network outputs to a ‘cleanup’ process to readout a selection signal. This might be done using a winner-take-all net, or by simply taking $\text{argmax}_j(z_j)$. The role of the fuzzy controller in this scheme is now to *bias* or influences this selection process rather than *govern* it ‘single-handed’.

While the mechanism of the top-down bias remains to be elucidated, the neural net pattern matching can be implemented in a straightforward way. We therefore attempted to do this using simple feedforward networks for the driving task shown in Figure 4a. The time series of the control signals (brake, steering, and throttle), are shown in Figure 4b. With only two controllers ($n = 2$), rather than use a 1-out-of-2 code, we used networks with a single output z with the interpretation that $z > 0.5$ signalled one controller rather than the other. Training was done using the Levenberg-Marquardt algorithm as implemented in the Matlab Neural Network toolbox.

In a first set of experiments, the three control signals formed the inputs, but in subsequent set, these were augmented with the signal derivatives, giving 6

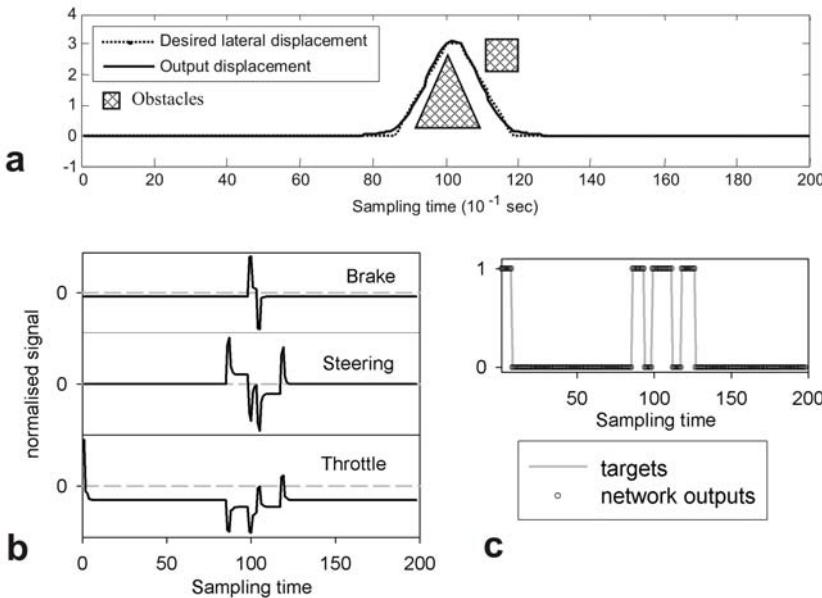


Fig. 4. Pilot study for training network matching in ‘habit-based learning’

inputs to each network. Best results were then obtained with 3 hidden nodes in a 6-3-1 network structure. A typical network output for the ‘brake’ network is shown in Figure 4c. The training worked well, with residual errors too small to see on the graph; this was also a feature of the networks for ‘steer’ and ‘throttle’. While it is encouraging to see highly accurate learning of the training set, the existence of good generalisation – certainly desirable if this scheme is to work robustly – requires the use of validation data. Gathering and deployment of this data are scheduled for future work, but in the interim, we note that the progressive reduction in layer size of the network is indicative that generalisation will, indeed, occur.

4 Discussion

Both psychological and neuroscientific thinking endorses the view that there are two quite different modes of behavioural control. Further, the psychological notions of automatic and controlled processing, would appear to have strong similarities with the neuroscientific counterparts of S-R (or habit-based) control, and goal-driven behaviour, respectively. We have suggested that adaptive controllers in an engineering context may benefit from deploying these two modes of control and reap benefits in terms of speed, and reduced errors during learning. We went on to show how at least part of this programme (automatised or S-R control) may be implemented in a particular AVC architecture.

However, many questions remain that are common to both biological and engineering domains. Studying a specific candidate for dual-process control (like the AVC architecture) therefore promises to be useful, not only for the particular application itself, but also in developing our understanding of automatic and controlled processing in the brain. Issues for the AVC controller include:

- Exactly how does the executive fuzzy controller bias the selection process? (One possible bridge between the two modes of control might make use of neuro-fuzzy hybrid techniques.)
- How do the habits get learned while having some degree of control? That is how can there be a seamless and sensible transition from controlled to automatised behaviour without slipping too quickly into poor ‘habitual’ behaviour?
- In solving this problem can we make use of the more natural ‘soft-selection’ seen in basal ganglia control? (‘Mixtures of controllers’)

The extent to which unique engineered solutions emerge will be forced to these questions, will strongly suggest their presence in the animal brain. We look forward to developing the programme of work outlined here and answering these questions in future work.

References

- [1] Styles, E.: *The Psychology of Attention*. Psychology Press, East Sussex (1997)
- [2] Packard, M.G., Knowlton, B.J.: Learning and memory functions of the basal ganglia. *Annu. Rev. Neurosci.* 25, 563–593 (2002)
- [3] Owen, A.M.: Cognitive planning in humans: neuropsychological, neuroanatomical and neuropharmacological perspectives. *Prog. Neurobiol.* 53(4), 431–450 (1997)
- [4] Abdullah, R., Hussain, A., Warwick, K., Zayed, A.: Autonomous intelligent cruise control using a novel multiple-controller framework incorporating fuzzy-logic based switching and tuning. *Neurocomputing* (in press)
- [5] Hussain, A., Gurney, K., Abdullah, R., Chambers, J.: Emergent common functional principles in control theory and the vertebrate brain: A case study with autonomous vehicle control. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 949–958. Springer, Heidelberg (2008)
- [6] Hommel, B., Ridderinkhof, K.R., Theeuwes, J.: Cognitive control of attention and action: issues and trends. *Psychol. Res.* 66(4), 215–219 (2002)
- [7] Schneider, W., Shiffrin, R.: Controlled and automatic information processing i. detection, search and attention. *Psychological Review* 84, 1–66 (1977)
- [8] Shiffrin, R., Schneider, W.: Controlled and automatic information processing ii. perception, learning, automatic attending and a general theory. *Psychological Review* 84, 128–190 (1977)
- [9] Neuman, O.: Automatic processing: A review of recent findings and a place for an old theory. In: Printz, W., Sanders, A. (eds.) *Cognition and Motor Processes*. Springer, Berlin (1984)
- [10] Birnboim, S.: The automatic and controlled information-processing dissociation: is it still relevant? *Neuropsychol. Rev.* 13(1), 19–31 (2003)

- [11] Graybiel, A.M.: Habits, rituals, and the evaluative brain. *Annu. Rev. Neurosci.* 31, 359–387 (2008)
- [12] Adams, C., Dickinson, A.: Instrumental responding following reinforcer devaluation. *Quarterly Journal of Experimental Psychology* 33, 109–122 (1981)
- [13] Norman, D., Shallice, T.: Attention to action: Willed and automatic control of behavior. In: Davidson, R.J., Schwartz, G.E., Shapiro, D. (eds.) *Consciousness and Self-Regulation*. Plenum Press, New York (1980)
- [14] Shallice, T.: Specific impairments of planning. *Philos. Trans. R Soc. Lond. B. Biol. Sci.* 298, 199–209 (1982)
- [15] Prescott, T.J., Bryson, J.J., Seth, A.K.: Introduction to the theme issue on modelling natural action selection. *Philos. Trans. R Soc. Lond. B. Biol. Sci.* 362, 1521–1529 (2007)
- [16] Garforth, J., McHale, S.L., Meehan, A.: Executive attention, task selection and attention-based learning in a neurally controlled simulated robot. *Neurocomputing* 69(16–18), 1923–1945 (2006)
- [17] Redgrave, P., Prescott, T.J., Gurney, K.N.: The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience* 89, 1009–1023 (1999)
- [18] Wilson, C.J.: Striatal circuitry: categorically selective, or selectively categorical? In: Miller, R., Wickens, J. (eds.) *Brain Dynamics and the Striatal Complex*. Volume submitted MS, pp. 289–306. Harwood Academic (1999)
- [19] Yin, H.H., Knowlton, B.J.: The role of the basal ganglia in habit formation. *Nat. Rev. Neurosci.* 7(6), 464–476 (2006)
- [20] Dalley, J.W., Cardinal, R.N., Robbins, T.W.: Prefrontal executive and cognitive functions in rodents: neural and neurochemical substrates. *Neurosci. Biobehav. Rev.* 28(7), 771–784 (2004)
- [21] Joel, D., Weiner, I.: The organization of the basal ganglia-thalamocortical circuits: open interconnected rather than closed segregated. *Neuroscience* 63, 363–379 (1994)
- [22] Haber, S.N., Fudge, J.L., McFarland, N.R.: Striatonigrostriatal pathways in primates form an ascending spiral from the shell to the dorsolateral striatum. *J. Neurosci.* 20(6), 2369–2382 (2000)
- [23] MacLeod, C.M.: Half a century of research on the stroop effect: an integrative review. *Psychol. Bull.* 109(2), 163–203 (1991)
- [24] Daw, N.D., Niv, Y., Dayan, P.: Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat. Neurosci.* 8(12), 1704–1711 (2005)
- [25] Conatser, R., Wagner, J., Ganta, S., Walker, I.: Diagnosis of automotive electronic throttle control systems. *Control Engineering Practice* 12, 23–30 (2004)
- [26] Lee, C.Y.: Adaptive control of a class of nonlinear systems using multiple parameter models. *Int. J. Contr. Autom. Sys.* 4(4), 428–437 (2006)

Multiple Sound Source Localisation in Reverberant Environments Inspired by the Auditory Midbrain

Jindong Liu^{1,*}, David Perez-Gonzalez², Adrian Rees²,
Harry Erwin¹, and Stefan Wermter¹

¹ Dept. of Computing and Technology,
University of Sunderland, Sunderland, SR6 0DD, United Kingdom

² Institute of Neuroscience, The Medical School,
Newcastle University, NE2 4HH, United Kingdom
{jindong.liu,harry.erwin,stefan.wermter}@sunderland.ac.uk,
{david.perez-gonzalez,adrian.rees}@newcastle.ac.uk
<http://www.his.sunderland.ac.uk>

Abstract. This paper proposes a spiking neural network (SNN) of the mammalian auditory midbrain to achieve binaural multiple sound source localisation. The network is inspired by neurophysiological studies on the organisation of binaural processing in the medial superior olive (MSO), lateral superior olive (LSO) and the inferior colliculus (IC) to achieve a sharp azimuthal localisation of sound sources over a wide frequency range in a reverberant environment. Three groups of artificial neurons are constructed to represent the neurons in the MSO, LSO and IC that are sensitive to interaural time difference (ITD), interaural level difference (ILD) and azimuth angle respectively. The ITD and ILD cues are combined in the IC to estimate the azimuth direction of a sound source. To deal with echo, we propose an inter-inhibited onset network in the IC, which can extract the azimuth information from the direct path sound and avoid the effects of reverberation. Experiments show that the proposed onset cell network can localise two sound sources efficiently taking into account the room reverberation.

Keywords: Spiking neural network, sound localisation, inferior colliculus, reverberation.

1 Introduction

Humans and other animals show a remarkable ability to localise multiple sound sources using the disparities in the sound waves received by the ears. For example, humans can localise as many as six concurrent sources [1] and cancel out echoes using two ears [2]. This has inspired researchers to develop new computational auditory models to help understand the biological mechanisms that underlie sound localisation in the brain. Binaural sound localisation systems take

* This work is supported by EPSRC (EP/D055466 and EP/D060648).

advantage of two important cues [3] derived from the sound signals arriving at the ears: (i) interaural time differences (ITD), and (ii) interaural level differences (ILD). Using these two cues sound source direction can be estimated in the horizontal plane.

In humans the ITD cue is effective for localising low frequency sounds ($20\text{ Hz} \sim 1.5\text{ kHz}$) [4], however, the information it provides becomes ambiguous for frequencies above $\sim 1\text{ kHz}$. In contrast, the ILD cue has limited utility for localising sounds below 1.5 kHz , but is more efficient than the ITD cue for frequencies higher than this [4]. The ITD and ILD cues are extracted in the medial and lateral nuclei of the superior olfactory complex (MSO and LSO), which project to the inferior colliculus (IC) in the midbrain. In the IC these cues are combined to produce an estimation of the azimuth of the sound [5]. The cells in the IC are classified into 6 main types among which onset and sustained-regular cells play the main role for sound azimuth detection even in an echo environment.

Several hypotheses and models for ITD and ILD processing have been proposed [3][6][7], with one of the most influential being a model advanced by Jeffress [3]. However, all above models only work in an anechoic environment. To deal with reverberation, Litovsky [8] proposed a model of the precedence effect which applies an onset detector to inhibit the localisation cues from the indirect sound path. Palomäki [9] simplified Litovsky's model by using envelope extraction. However, these models did not exploit the biological pathways from the MSO/LSO to the IC, such as the inhibition from the ipsilateral LSO to the IC. These pathways are believed the crucial key for a sharp localisation over broadband frequency.

This paper presents a model designed to identify multiple sound source directions by means of a spiking neural network (SNN). It is the first to employ a SNN that combines both ITD and ILD cues derived from the SOC in a model of the IC to cover a wide frequency range and to target a reverberant environment. This model incorporates biological evidence on the inputs from the MSO and LSO to the IC, and is able to build a sharp spatial representation of sound source. To cope with an reverberant environment, onset cells in the IC are modelled and interconnected to each other by inhibition projection.

2 Biological Fundamentals and Assumptions

When sound waves arrive at ears, the temporal and amplitude information is encoded and transmitted to the MSO and LSO in order to extract ITDs and ILDs respectively [5]. According to Jeffress's original model, the ITD-sensitive cells in the MSO can be idealised as a coincidence cell array where each cell receives a delay-line input, and the cells are assumed to be distributed along two dimensions: CF and ITD (see Figure 1). A cell in the MSO fires when the contralateral excitatory input leads the ipsilateral by a specific time difference.

For the LSO model, we represent the cells in the LSO distributed across two dimensions, CF and ILD, in an analogous manner to the MSO (Figure 2), but without any interaural delay. Each LSO cell compares the input levels from two

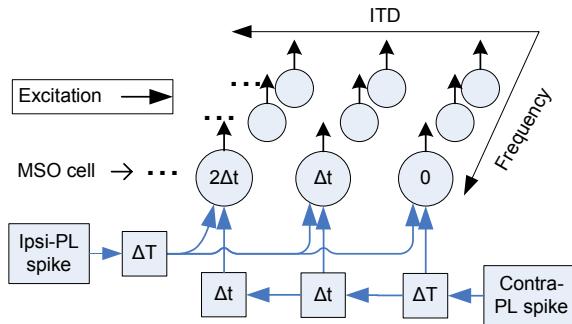


Fig. 1. Schematic diagram of the MSO model. While all the spike trains from the ipsilateral side share the same delay, the ones originating from the contralateral side are subjected to variable delays. The difference between the ipsilateral and contralateral delays makes each cell in the MSO model most sensitive to a specific ITD.

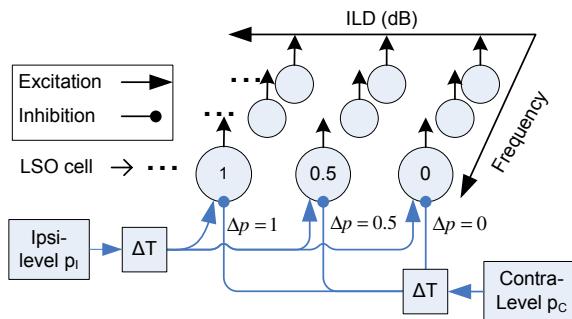


Fig. 2. Schematic diagram of the LSO model used in our system. Analogous to Figure 1 we assume that there are cells most sensitive to a given ILD and frequency. When the exact level difference $\Delta p = \log(\frac{p_I}{p_C})$ is detected, the corresponding LSO cell fires a spike.

ears and generates a spike if the level difference is equal to the characteristic ILD of the cell.

All the outputs of the MSO and LSO are projected to the inferior colliculus (IC). The IC is also tonotopically organised, and contains a series of iso-frequency laminae, which span the whole range of frequencies perceived by the animal. In this model, we assume for simplicity that there are only projections from cells with the same CF. Consequently in our model the laminae of the IC with low CF (200 Hz to 1 kHz) only receive projections from the MSO, while the laminae with higher frequencies (up to 4 kHz) receive projections from both the MSO and LSO. The laminae with a CF above 4 kHz would only receive inputs from the LSO, but our model does not include this range of frequencies.

The cells in the IC can be classified into 6 physiological types [10]: sustained-regular, rebound-regular, onset, rebound-adapting, pause/build and rebound-transient. The sustained-regular cells generate regular spikes when their input is

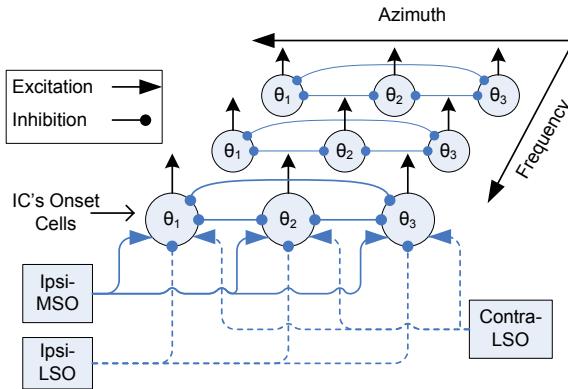


Fig. 3. Schematic diagram of the IC model. In the range of 20 Hz to 1 kHz, the IC model only receives inputs from the MSO model (solid line). From 1 kHz to 4 kHz, the IC model receives inputs from the MSO model (solid line) and both LSOs (dash line).

kept positive and is can detect ongoing sounds. We hypothesise that these cells could encode sound source locations in the free field in the absence of echoes. However, in a reverberant environment, an echo is added to the sound taking the direct path and this causes a detection error in the output of sustained-regular cells. In contrast, onset cells only generate one spike when the input current changes from 0 to positive and then cease firing as long as the input is kept positive. This property would be useful when the IC model is locating a sound in an echoic environment, because its output spike is only related to the sound taking the direct path.

Taking into account this biological evidence, we propose an IC model for sound source localisation as outlined in Figure 3. Analogous to the biology evidence, the IC model consists of different components according to the frequency domain. In addition, we suppose that onset cells with the same CF suppress one another, i.e. an early spike from one onset cell will inhibit other onset cells which have the same characteristic frequency. We will describe this inhibitory network in detail in the next section.

3 System Model of Sound Localisation

Inspired by the neurophysiological findings and the proposed models presented in Section 2, we designed our model to employ spiking neural networks (SNNs) that explicitly take into account the timing of inputs and mimic real neurons. The cues used for sound localisation, such as time and sound level, are encoded into spike-firing patterns that propagate through the network to extract ITD and ILD and calculate azimuth. Every neuron in the SNN is modelled using a leaky integrate-and-fire (LIF) model. The response of a neuron to spike inputs is modelled by:

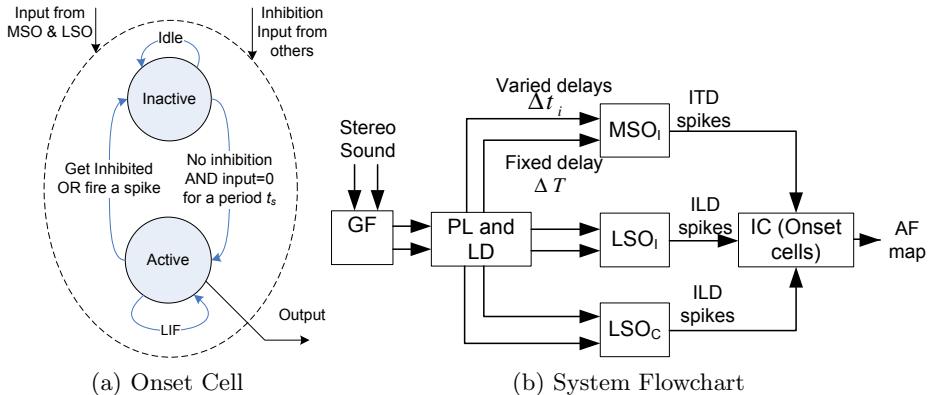


Fig. 4. (a) The IC's onset cell. (b) Flowchart of the biologically inspired sound localisation system. This example only shows one IC; note that there is a symmetric model for the contralateral IC. MSO_I ipsilateral MSO model; LSO_I ipsilateral LSO model; LSO_C contralateral LSO model; GF Gammatone filterbank; PL phase locking; LD level detection; SR sustained-regular and AF azimuth-frequency.

$$C \frac{du}{dt} = \sum_k I_k(t) - \frac{C}{\tau_m} u \quad (1)$$

$$t_f : u(t_f) = \phi$$

where $u(t)$ is the membrane potential of the neuron relative to the resting potential which is initialised to 0, and τ_m is a time constant. C is the capacitance which is charged by $\sum_k I_k(t)$ from multiple inputs, where $I_k(t)$ is a current input which is a constant square with amplitude w_s and duration τ_s in response to a spike input. k is the number of input connections to the neuron. The action potential threshold φ controls the firing time t_f . When $u(t) = \varphi$, the soma will fire a spike; then $u(t)$ is reset to 0. Afterwards, the soma will be refractory for a period $t_r = 1$ ms during which it will not respond to any synaptic inputs. After the refractory period, the soma returns to the resting state.

The LIF model can be used to represent the cells in the MSO and the LSO and the sustained-regular cell in the IC. However, the LIF model cannot directly model the onset cell because it constantly responds to continuous inputs, rather than just the initial onset input. Instead, for the onset cell, we propose a hybrid model of LIF and a state machine. Each onset cell has two states: active and inactive. When the cell is active, the cell is implemented as a LIF neuron until a spike is fired, or the cell receives an inhibitory input, after which the cell's state becomes inactive. The cell goes back to active state only if there is no inhibition and the input is 0 (no spike) for a period t_s (see Figure 4a).

A schematic structure for the sound localisation procedure is shown in Figure 4b. The frequency separation occurring in the cochlea is simulated by a bandpass filterbank consisting of 16 discrete second-order Gammatone filters [II], that produce 16 frequency bands between 200Hz and 4kHz. After the Gammatone filterbank, the temporal information in the waveform in each frequency channel

is encoded into a spike train by the phase locking module in Figure 4b. Every positive peak in the waveform triggers a phase-locked spike to feed into the MSO model. The sound level is detected in the same module but directed to the LSO model.

To calculate the ITD, the phase-locked spike trains are then fed into the MSO model. A series of delays are added to the spike trains of the contralateral ear to simulate the delay lines Δt_i (see Figure 1). The spike train of the ipsilateral ear reaches the MSO with a single fixed delay time ΔT . The parameters of cells in the MSO are set: $l_s = 2.1\text{ms}$, $\tau_s = 0.08\text{ms}$, $\tau_m = 1.6\text{ms}$, $\varphi = 8\text{e-}4$, $w_s = 0.1\text{A}$ and $C = 10\text{mF}$.

The ILD pathway is not modelled using a LIF model; rather the sound levels previously detected for each side are compared and the level difference is calculated. The LSO model contains an array of cells distributed along the dimensions of frequency and ILD (Figure 2). When a specific ILD is detected at a given frequency, the corresponding LSO cell fires a spike. The level difference is calculated as $\Delta p^j = \log(p_I^j/p_C^j)$, where p_I^j and p_C^j stand for the ipsilateral and contralateral sound pressure level for the frequency channel j .

After the basic cues for sound localisation have been extracted by the MSO and LSO models, the ITD and ILD spikes are fed into the IC model, as shown in Figure 4b. The IC model merges the information to obtain a spatial representation of the azimuth of the sound source. According to the model proposed in Section 2 we need to define the connection strength between the ITD-sensitive cells (m_i) in the MSO and the azimuth-sensitive cells (θ_j) in the IC, and the connection between the ILD-sensitive cells (l_i) in the LSO and θ_j . In a SNN, each of the inputs to a neuron (in this case in the IC) produces a post-synaptic current $I(t)$ in the modelled cell. The post-synaptic currents of all the inputs are integrated to calculate the response of the neuron. To modify the weight of each input we assign a different gain to the amplitude w_s of the post-synaptic current $I(t)$ (in Equation 1) of each connection. Inspired by Willert's work [6], we used an approach based on conditional probability to calculate these gains, as shown in the following functions:

$$e_{m_i\theta_j} = \begin{cases} p(\theta_j | m_i, f) & \text{if } p > 0.5 \max_j(p(\theta_j | m_i, f)) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$e_{l_i\theta_j} = \begin{cases} p(\theta_j | l_i, f) & \text{if } p > 0.8 \max_j(p(\theta_j | l_i, f)), f \geq f_b \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$c_{l_i\theta_j} = \begin{cases} 1 - p(\theta_j | l_i, f) & \text{if } p < 0.6 \max_j(p(\theta_j | l_i, f)), f \geq f_b \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

where $e_{m_i\theta_j}$ and $e_{l_i\theta_j}$ represent the gain of the excitatory synapse between the MSO and LSO respectively and the IC. If $e_{m_i\theta_j} = 0$, it is equivalent to no connection between m_i and θ_j . Similarly, $e_{l_i\theta_j} = 0$ indicates no connection between l_i and θ_j . The term f_b is the frequency limit between the low and middle frequency regions and is governed by the separation of the ears and the dimensions

of the head of the “listener”. Based on the dimensions of the robot head used in this study, f_b should be around 850Hz.

$c_{l_i \theta_j}$ represents the gain of the inhibitory synapse between the LSO and the IC. f stands for the centre frequency of each frequency band. $p(*)$ stands for a conditional probability, which can be calculated by Bayesian probability and $p(\theta_j | m_i, f)$ can be calculated by:

$$p(\theta_j | m_i, f) = \frac{p(m_i | \theta_j, f)p(\theta_j | f)}{p(m_i | f)} \quad (5)$$

In a physical model, the conditional probability $p(m_i | \theta_j, f)$ is obtained from the statistics of sounds with known azimuths. To obtain such data, we recorded a 1s-sample of white noise sounds coming from 37 discrete azimuth angles (from -90 to 90 degrees in 5 degree steps) using a robot head. The head had dimensions similar to an adult human head and included a pair of cardioid microphones (Core Sound) placed at the position of the ears, 15 cm apart from one another.¹

These recordings were processed through our MSO model to obtain an ITD distribution for each azimuth, which was then used to calculate $p(m_i | \theta_j, f)$. Finally, we applied Equation 5 to Equation 2 to calculate the gain, $e_{m_i \theta_j}$, of the connection between the MSO cells and the IC cells. These gains are further adjusted to leave only components consistent with the known anatomy of the pathway, i.e. there is no significant projection from the contralateral MSO to the IC. A similar procedure is used to calculate the gains of the LSO projection to the IC.

Equations 2 and 3 map the excitatory connections of each MSO and LSO cell to the IC cells representing the most likely azimuths, while Equation 4 maps the inhibitory LSO projection to cells representing azimuths in the hemifield opposite to the sound source. This inhibition counteracts the effects of false ITD detection at high frequencies.

4 Experimental Results

In this section, we first verify our model by locating a pure tone in a reverberant environment. We then implement our model to locate three groups of two concurrent sound sources in the same environment, and compare the results using sustained regular cells with those using onset cells.

Figure 5a compares the localisation results obtained using onset and sustained-regular cells. The sound sample was a 700Hz pure tone played at 10 degrees in a reverberant environment (echo delay 6 ms). The possibility of sound source azimuth is calculated as the division of the number of spikes of the IC cells in one

¹ Sounds were recorded in a low noise environment with 5 dB SPL background noise. The distance of the sound source to the center of the robot head was 128 cm and the speakers adjusted to produce 90 ± 5 dB SPL at 1 kHz. Recordings were digitalised at a sample rate of 44100 Hz. Sound duration was 1.5s, with 10 ms of silence at the beginning.

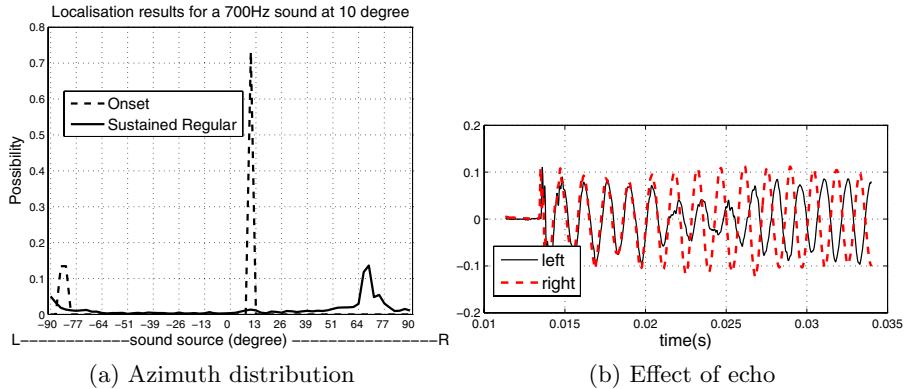


Fig. 5. (a) Azimuth distribution of single sound localisation in a reverberant environment. (b) The effect of room reverberation to the recording. The sound sample is a 700Hz pure tone played at 10 degree from the midline.

azimuth angle of all frequency channels by the total number of spikes. Two methods were tried for the same sound: (i) only using sustained-regular cells in the IC with no inhibitory connection between the cells, and (ii) use the onset cells in the IC and the inhibitory network proposed earlier in this paper. Note that the first method is equivalent to the most conventional methods of sound localisation which are based on ongoing sound detection. Figure shows that the result from onset cells has a peak around 12 degrees which is very close to the real sound azimuth, while the peak of the results from the sustained-regular cells is around 70 degrees which is far from the true location. The main reason is that the reverberation interfered with the sound wave reaching the microphone and changed the ITD and ILD cues.

Figure 5b shows the effect of room reverberation on the recordings of a 700Hz pure tone presented at the front. It shows that the sound reaches the microphone at 0.013s and the signals from both microphones match each other peak by peak. However, from about 0.02s, the reverberant sound arrived and started to interfere with the recorded sound. As a result, the signals shifted by about 0.007ms. This peak shifting caused the localisation error that occurred when using the modelled sustained-regular cells.

To test our model for a mixture of two concurrent sound sources, we designed three test groups and used a 500Hz pure tone from -90 degree as the first sound source for all three groups. The second source sources in three groups were designed as speech “hello”, “coffee” or white noise. The second sound source is presented from 7 positions from -90 to 90 degree for every 30 degrees. All the sound sources are 1.28m far from the robot head. During recording, two sound sources are played at the same time. Each test point includes 10 sound sets and the final results are the average value. Figure 6a shows the localisation results using sustained-regular cells. In the figure, the second sound source azimuths were calculated accurately, however the results for the first sound source show a

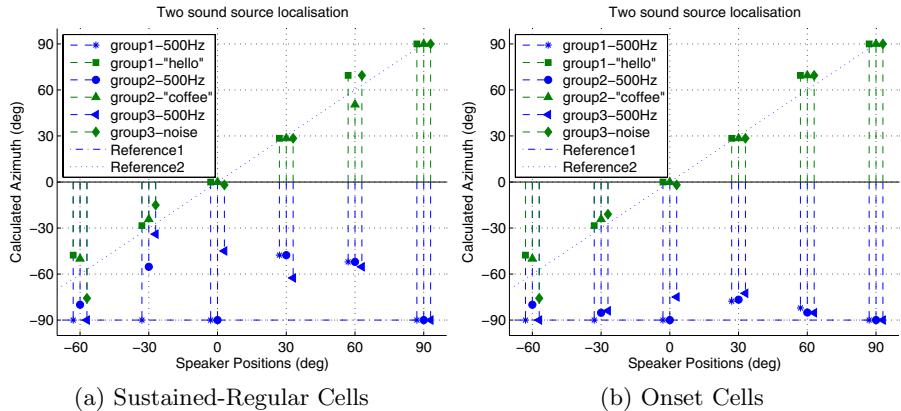


Fig. 6. Sound localisation results for three groups of concurrent sound sources. Reference1 stands for the ideal azimuth of first sound source 500Hz and reference2 for the second sound source's ideal azimuth.

big offset from the ideal detection. In contrast, Figure 6b shows more accurate localisation results in both sound sources.

5 Conclusion and Future Work

This paper describes the design and implementation of a sound localisation model that uses a SNN inspired by the mammalian auditory system for a reverberant environment. In this system, both ITD and ILD pathways were modelled and computed in the MSO and LSO models, and the ITD spike and ILD spikes were projected to the IC in a way similar to the biological system where they were merged together to achieve broadband sound localisation. Onset IC cells are modelled and an inhibitory onset network is proposed to eliminate the echo. The experimental results showed that our system can localise two concurrent sound sources in a reverberant environment especially for pure tones with azimuths between -90 and 90 degrees. Our model's success casts light on the mobile robot application in real world application where reverberation is unavoidable. In the future, other IC cell types will be tested in the model. For the application of our system to a mobile robot, we plan to implement a sound separation system based on sound source direction in order to improve speech recognition in a noisy environment.

References

1. Bronkhorst, A., Plomp, R.: Effect of multiple speechlike maskers on binaural speech recognition in normal and impaired hearing. *The Journal of the Acoustical Society of America* 92, 3132–3139 (1992)

2. Blauert, J.: Spatial Hearing: The Psychophysics of Human Sound Localization. MIT Press, Cambridge (1997)
3. Jeffress, L.: A place theory of sound localization. *J. Comp. Physiol. Psychol.* 41, 35–39 (1948)
4. Moore, B.: An Introduction to the Psychology of Hearing. Academic Press, San Diego (2003)
5. Yin, T.: Neural mechanisms of encoding binaural localization cues in the auditory brainstem. *Integrative Functions in the Mammalian Auditory Pathway I*, 99–159 (2002)
6. Willert, V., Eggert, J., Adamy, J., Stahl, R., Koerner, E.: A probabilistic model for binaural sound localization. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* 36(5), 982–994 (2006)
7. Voutsas, K., Adamy, J.: A biologically inspired spiking neural network for sound source lateralization. *IEEE Trans. Neural Networks* 18(6), 1785–1799 (2007)
8. Litovsky, R., Colburn, H., Yost, W., Guzman, S.: The precedence effect. *Journal of the Acoustical Society of America* 106(4 I), 1633–1654 (1999)
9. Palomäki, K., Brown, G., Wang, D.: A binaural processor for missing data speech recognition in the presence of noise and small-room reverberation. *Speech Communication* 43(4), 361–378 (2004)
10. Sivaramakrishnan, S., Oliver, D.: Distinct K Currents Result in Physiologically Distinct Cell Types in the Inferior Colliculus of the Rat. *Journal of Neuroscience* 21(8), 2861 (2001)
11. Slaney, M.: An efficient implementation of the patterson-holdsworth auditory filter bank. Apple Computer Technical Report 35 (1993)

A Model of Neuronal Specialization Using Hebbian Policy-Gradient with “Slow” Noise

Emmanuel Daucé^{1,2}

¹ INRIA Lille Nord-Europe, Villeneuve d'Ascq, France

² Institute of Movement Sciences, University of the Mediterranean, Marseille, France

Abstract. We study a model of neuronal specialization using a policy gradient reinforcement approach. (1) The neurons stochastically fire according to their synaptic input plus a noise term; (2) The environment is a closed-loop system composed of a rotating eye and a visual punctual target; (3) The network is composed of a foveated retina, a primary layer and a motoneuron layer; (4) The reward depends on the distance between the subjective target position and the fovea and (5) the weight update depends on a Hebbian trace defined according to a policy gradient principle. In order to take into account the mismatch between neuronal and environmental integration times, we distort the firing probability with a “pink noise” term whose autocorrelation is of the order of 100 ms, so that the firing probability is overestimated (or underestimated) for about 100 ms periods. The rewards occurring meanwhile assess the “value” of those elementary shifts, and modify the firing probability accordingly. Every motoneuron being associated to a particular angular direction, we test at the end of the learning process the preferred output of the visual cells. We find that accordingly with the observed final behavior, the visual cells preferentially excite the motoneurons heading in the opposite angular direction.

1 Introduction

The issue of plasticity and neuronal specialization is important in computational neuroscience, in order to explain more realistically the adaptation processes taking place in the brain. When looking at the first steps of sensory processing, for instance, the topology of the captors appears prominent in the shaping of the cortical maps (“somatotopic” layers, “retinotopic” layers, “tonotopic” layers, etc...). This topography is expected to be the result of a progressive plasticity process which shapes the input/output response curve of a population of neurons according to their location, making them more reactive to a particular class of signals. Similar adaptation processes are expected to take place at all the stages of the brain processing, continuously modifying the way the brain will take its decisions and control its movements.

Taking now this problem under a reinforcement learning (RL) framework, a brain is seen as a neural controller facing a complex decision problem, receiving non-stationary perceptions and continuously sending motor commands which immediately take effect in the sensory scene. The relevance of the motor commands is not known, and some

signals from the environment (or from the body) indicate to which point the choices made are “good” or “bad”. The controller being composed of a huge population of neurons, the global decision is the combination of many local infinitesimal decisions (i.e. emitting a spike or not at time t). A local process that would appropriately shape the neuronal decision according to the reward is expected to favour better collective responses, thus improving the performances of the controller.

2 Principles

Classical RL approaches consider value functions which map some (state, action) couples to a value estimating the expectation of future rewards [1]. In the case of a parametrized stochastic policy implemented by binary units, a policy gradient approach [2] avoids such explicit value function and directly shapes the parameters according a local estimation of the gradient of the reward expectation. It ends up as an expression where the weights update depends on the Hebb-like product $[S(t) - f(I(t))] \times \varepsilon_j(t)$ where S is the actual neuron output, f is the firing probability according to the input and ε_j is the pre-synaptic activity. The global process is a gradient ascent algorithm. Bartlett and Baxter [3] show that it can be extended to any network of stochastic neurons where the reward signal is spread to every neuron. In that case, the policy improvements realized locally participate to the improvement of the global response.

Some extensions to more realistic spiking neurons have been proposed : Seung [4] treated the case of stochastic synapses with poisson firing. Baras [5] considers a Spike Response Model (SRM) [6] with a stochastic spike emission mechanism (proportional to the weighted sum of pre-synaptic activities). Florian [7] considers the more classical case of stochastic SRM neurons with escape noise, applied to a closed-loop reinforcement task. In all of those models the characteristics of the noise is mixed with the model of spike emission.

The purpose of our model is to prove that such neuro-realistic implementations of very general gradient ascent principles can reproduce the emergence of global regularities, as observed in biological neural networks. We use in the following a model which shares some similarities with [7], but we put a stronger emphasis on the characteristics of the noise, in order to improve the learning process.

Several temporal scales are indeed to be considered when modeling such neuromimetic controllers. First, the typical integration time of the neurons is of the order of few milliseconds. Second, the motor commands have a duration of the order of 100 ms. In the design of an adaptive controller, this temporal mismatch must be taken into account, so that the process will be guided by signals containing some random “events” whose typical duration will be of the same order than the events taking place in the environment to be controlled. For that, we consider that the firing probability is driven by a “pink noise” term whose autocorrelation is of the order of 100 ms, so that the firing probability is overestimated (or underestimated) for periods of about 100 ms. The rewards occurring meanwhile will assess the “value” of those elementary shifts, and modify the firing probability accordingly.

3 Model

3.1 Neuron Model

We consider a simple model of spiking neurons. A neuron is defined by a set of N synaptic weights $W = \{w_1, \dots, w_N\}$. The neuron synaptic input is

$$I_{syn}(t) = \sum_j w_j \varepsilon_j(t - d_j)$$

where $\varepsilon_j(t)$ is the post-synaptic potential (PSP) of the j -th neuron and d_j is the axonal delay. We use a discretized Spike Response Model approach [6] for modelling the PSP arrivals and membrane refractoriness. We take here a classical exponential PSP kernel, i.e.

$$\varepsilon_j(t) = \sum_{s < t, S_j(s)=1} \exp\left(-\frac{t-s}{\tau_m}\right)$$

where τ_m is the membrane time constant.

According to the “escape noise” principle [6], the noise source is decoupled from the spike emission process, i.e. is modelled as an independent input $I_{noise}(t)$. It may be considered as an additional signal coming from an external source (a supplementary layer of neurons for instance). The total neuronal input is then $I(t) = I_{syn}(t) + I_{noise}(t)$. The membrane potential is $V(t) = I(t) - \eta(t)$ with refractoriness

$$\eta(t) = \sum_{s < t, S(s)=1} V(s) \exp\left(-\frac{t-s}{\tau_m}\right)$$

There is no stochasticity in the spike emission mechanism: a spike is emitted at t if $V(t) \geq \theta$ and if the previous spike was emitted before $t - \tau_r$, where τ_r is the refractory period. The output of the neuron $S(t)$ is 0 (no spike) or 1 (spike).

3.2 Firing Probability

The firing probability is $f = P(S(t) = 1 | I_{syn}(t))$, and is expected to depend on the noise and δt . This hypothesis holds if we estimate the firing rate on large time interval $T \gg \tau_m$, and ignore some fluctuations of the probability related to the refractoriness that follows the spike emission. The neuron firing rate is $\nu = 1/\langle ISI \rangle$ where $\langle ISI \rangle$ is the mean inter-spike interval. Then, we define the firing probability in a small temporal interval $[t, t + \delta t]$ as $f = \nu \delta t$ (where $\delta t < \tau_r = \frac{1}{\nu_{max}}$). We suppose in the following that a relationship can be established between the total synaptic input I_{syn} and the firing probability. The firing probability f can be given explicitly on the basis of an estimation of mean inter-spike interval, i.e. $f(I_{syn}) = \frac{\delta t}{\langle ISI(I_{syn}) \rangle}$ (see for instance [68]). The explicit calculation being rather intricate, we provisionally use an estimation of f on the basis of a set of empirical measures of $(I_{syn}(t), S(t))$ couples.

3.3 Gradient Ascent Principles

A reward r can occur at time t , depending on the neuron input/output $(\mathcal{E}(t), S(t))$ where \mathcal{E} is the pre-synaptic activity. The expectation of the reward is thus supposed to rely on

the parameters of the input/output mapping, i.e. on the neuron's weights \mathcal{W} . We note $J(\mathcal{W}) = \mathbb{E}(r(t))$. A local estimator of the gradient of $J(\mathcal{W})$ can be given on the basis of the neuron input $I_{syn}(t)$, using a likelihood ratio approach [3], namely

$$\nabla_{\mathcal{W}} J = E \left[r(t) \times \left(S(t) \frac{\nabla_{\mathcal{W}} f(I_{syn}(t))}{f(I_{syn}(t))} + (1 - S(t)) \frac{\nabla_{\mathcal{W}} (1 - f(I_{syn}(t)))}{1 - f(I_{syn}(t))} \right) \right] \quad (1)$$

This estimator can be used to modify the weights at each time step in order to increase the expectation of the reward. In the case of a temporal credit assignment problem with a delayed reward occurring at time T , a local estimate of this gradient is $\nabla_{\mathcal{W}} J \simeq r(T)Z(T)$ [2] with

$$Z(t) = \lambda Z(t - \delta t) + (1 - \lambda) \left(S(t) \frac{\nabla_{\mathcal{W}} f(I_{syn}(t))}{f(I_{syn}(t))} + (1 - S(t)) \frac{\nabla_{\mathcal{W}} (1 - f(I_{syn}(t)))}{1 - f(I_{syn}(t))} \right) \quad (2)$$

with $\lambda \in [0, 1]$. By analogy with classical reinforcement learning techniques [9], $Z(t)$ is called the eligibility trace.

3.4 Weights Update

Two cases can be considered.

Dense Rewards. When the reward signal is present at each time step of the process, the weight update is

$$w_j(t) = w_j(t - \delta t) + \alpha r(t) \Phi(t) \times (S(t) - f(I_{syn}(t))) \times \varepsilon_j(t) \quad (3)$$

which is deduced from (1), using the learning parameter α , with factor

$$\Phi(t) = \frac{f'(I_{syn}(t))}{f(I_{syn}(t))(1 - f(I_{syn}(t)))}$$

This update rule only relies on local quantities available at the vicinity of the synapse, which makes it applicable for the modelling of biologically inspired learning processes. Its main advantage is its applicability to large networks of neurons where the same reward is sent to every neuron.

So the weight change is based on an estimate of the difference between the actual firing $S(t)$ and the firing probability according to the synaptic input $f(I_{syn}(t))$. When no noise modifies the actual firing, the response may not be improved in time since the two quantities are expected to cancel each other. The noise introduces a distortion in the firing rate, so that the expectation of the difference can be positive (resp. negative) if the random signal locally increases (resp. decreases) the actual firing rate. The reward taking place meanwhile will validate (or invalidate) this distortion and modify the weights accordingly, favouring higher (resp. lower) firing rates in case of positive reward.

This rule shares common characteristics with the classical Hebbian rule [10], and the first term of the product is typically Hebbian. The subtracted term corresponds to a slow decay occurring when no spike happen. When the noise is centered, the rule is

"balanced" i.e. the expectation of weight change is zero and no weight drift is expected to happen, either for positive or negative rewards. This point is important since it simplifies the design of the reward, as no particular balance between positive and negative rewards needs to be considered.

Sparse Rewards. In the case of occasional rewards, the weights update rule expression is:

$$\begin{aligned} w_j(t) &= w_j(t - \delta t) + \alpha r(t) z_j(t) \\ z_j(t) &= \left(1 - \frac{\delta t}{\tau_z}\right) z_j(t - \delta t) + \frac{\delta t}{\tau_z} Z_j(t) \\ Z_j(t) &= \Phi(t) \times (S(t) - f(I_{syn}(t))) \times \varepsilon_j(t) \end{aligned} \quad (4)$$

where τ_z is the trace time constant. The larger τ_z , the larger the time interval taken into account for the gradient estimate. If $\tau_z = \delta t$, we go back to equation (3). When the rewards are rare events, the trace constants needs to be large for taking into account the full sequence of actions that have led to them. Occasional reward correspond to a situation frequently encountered in real control problems. The existence of a physiological equivalent of this trace is plausible since it only relies on quantities which are locally available at the synapse.

3.5 Self-correlated Noise

The capacity to avoid local minima crucially depends on the quality of the exploration process and thus on the characteristics of the noise injected in the system.

In order to help the learning process to catch the effect of a particular shift from the expected firing rate, we use a self-correlated Gaussian noise whose leak is τ_{noise} :

$$I_{noise}(t) \sim \left(1 - \frac{\delta t}{\tau_{noise}}\right) I_{noise}(t - \delta t) + \sqrt{2 \frac{\delta t}{\tau_{noise}} - \left(\frac{\delta t}{\tau_{noise}}\right)^2} \times \mathcal{N}(0, \sigma_{noise}^2)$$

This noise model is centered, gaussian with standard deviation σ_{noise} for long durations, but remains correlated on a $\pm\tau_{noise}$ ms window. In that case, on the contrary to simple escape noise or diffusive noise models [6], the deviations from the expected firing rate lasts far longer than the membrane time constant.

3.6 Neuronal Controller

A neuronal adaptive controller is a set of interconnected neurons where a subpopulation of neurons is driven by the sensory signal, and the activity of another subpopulation is used to define the motor command(s). The neurons which receive the sensory signal are primary sensory neurons and the ones that drive the motor command velocity are pre-motor neurons. If we consider a system composed of P populations, every population owns $N^{(p)}$ neurons, and the input of one particular neuron is now:

$$I_i^{(p)}(t) = I_{i,syn}^{(p)}(t) + I_{i,noise}^{(p)}(t)$$

with

$$I_{i,syn}^{(p)}(t) = I_{i,ext}^{(p)}(t) + \sum_{q=1}^P \sum_{j=1}^{N^{(q)}} w_{ij}^{(pq)} \varepsilon_j^{(q)}(t - d_{ij}^{(q)})$$

A reward r is sent (densely or occasionally) to every neuron. The neurons locally modify their synaptic weights in order to maximize the expectation of reward. Bartlett and Baxter have shown [3] that a combination of local optimizations drives the system toward a global improvement of the actions.

4 Application

We consider a neural network controller embedded in a virtual environment. The example taken here is a model of a visual tracking system. It has no direct biological counterpart, but owns some characteristics (foveated retina, continuous moves, simple reward) that makes him suitable in a perspective of biological modelling. The achievement of the task (target tracking) is not a challenge in itself. Our aim is to study how the system learns some regularities from the environment on the basis of simple global constraints.

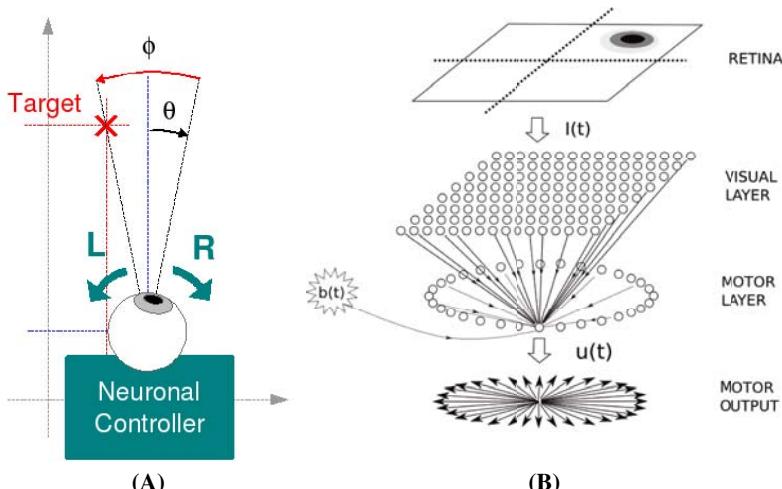


Fig. 1. (A) Environment: A rotating eyeball is tracking a punctual target (cross): θ is the actual eye direction, ϕ is the subjective target direction (relative to the pupil's direction). **(B) Controller:** The visual layer is composed of 256 neurons sending excitatory axons toward a motor layer composed of 32 neurons. The premotor neurons inhibit each other. A slow noise $b(t)$ is added to every premotor neuron. $I(t)$ is the visual signal, and $u(t)$ is the command (see text for more details).

4.1 Environment

Consider a fixed eyeball with 2 degrees of freedom in rotation. The current direction of the pupil is given by $\overrightarrow{\theta(t)} = (\theta_x(t), \theta_y(t))$. A target is moving in the visual field and the controller must track the target by making it as close as possible to the fovea. Its direction is $\overrightarrow{\psi(t)} = (\psi_x(t), \psi_y(t))$. Its subjective direction (i.e. direction relative to the pupil) is $\overrightarrow{\phi(t)} = \overrightarrow{\psi(t)} - \overrightarrow{\theta(t)} \in [-\pi, \pi]^2$. The movement of the eye is controlled by $\overrightarrow{u(t)} = (\dot{\theta}_x(t), \dot{\theta}_y(t))$. In the examples that follow, the target is moving circularly in front of the eyeball. Its objective direction is $\overrightarrow{\psi(t)} = (\cos(\frac{2\pi t}{320}), \sin(\frac{2\pi t}{320}))$ whose period is 320 ms.

4.2 Controller

The controller is composed of a retina and 2 layers of neurons. We use a 2D foveated retina composed of $N^{(1)} = 16 \times 16 = 256$ pixels. The spatial resolution is stronger at the center of the retina than at the periphery (like in vertebrate eyes). The perceived direction is proportional to the log of the real distance to the center:

$$\overrightarrow{d(t)} = \text{sign}(\overrightarrow{\phi(t)}) \times \log_{10}(1 + 9 \frac{|\overrightarrow{\phi(t)}|}{\pi}) \in [-1, 1]^2$$

The first layer consists of 256 neurons which are directly stimulated by the external signal. Consider that each cell $i \in \{1, \dots, N^{(1)}\}$ of the first layer responds to a preferred input direction $\overrightarrow{d_i}$. Taking $M = \sqrt{N^{(1)}}$, $k = \lfloor \frac{i-1}{M} \rfloor + 1$, $\ell = i - (k-1)M$, so that $(k, \ell) \in \{1, \dots, M\}^2$, we fix $\overrightarrow{d_i} = (\frac{2k-(M+1)}{M}, \frac{2\ell-(M+1)}{M})$. Then

$$I_{i,ext}^{(1)}(t) = 2 \left\langle \cos((\overrightarrow{d(t)} - \overrightarrow{d_i}) \times \pi) \mid H(\cos((\overrightarrow{d(t)} - \overrightarrow{d_i}) \times \pi)) \right\rangle$$

where $\langle . | . \rangle$ is the inner product and H is the Heaviside function.

The second layer consists of $N^{(2)} = 32$ neurons. It sends a velocity command $\overrightarrow{u(t)} = (\dot{\theta}_x, \dot{\theta}_y)$ to the eye effectors. Every neuron of the second layer is associated with a particular angular direction, i.e. $\theta_i = i \times \frac{\pi}{N^{(2)}}$ rad. The output of the controller is defined according to the spiking activity of the population, i.e.

$$\overrightarrow{u_i(t)} = \varepsilon_i^{(2)}(t) \times (\sin(\theta_i), \cos(\theta_i))$$

and

$$\overrightarrow{u(t)} = \frac{1}{\sqrt{N^{(2)}}} \sum_{i=1}^{N^{(2)}} \overrightarrow{u_i(t)}$$

which is normalized in $\sqrt{N^{(2)}}$ in order to maintain the variance of the command independent of $N^{(2)}$.

The neurons of the first layer send excitatory links to the second layer. The neurons of the second layer send lateral inhibitory links so that the global input of the premotor neurons is balanced. The connectivity is initially random. Considering the synaptic

sums $W_i^{(pq)} = \sum_{j=1}^{N^{(q)}} w_{ij}^{(pq)}$, the weights are randomly set so that $E(W_i^{(pq)}) = \mu^{(pq)}$ and $\text{var}(W_i^{(pq)}) = \sigma^{(pq)2}$. Here we fix $\mu^{(21)} = 2.5$, $\sigma^{(21)} = 0.1$, $\mu^{(22)} = -2.5$, $\sigma^{(22)} = 0.25$.

A slow noise is added to the input of the premotor neurons. If not precised, we use in the following $\tau_{noise} = 100$ ms and $\sigma_{noise} = 0.35$.

4.3 Reward

We expect the system to learn to compensate the visual error by making a movement in the observed direction of the target, i.e. to learn the appropriate feedback homeostatic controller. The reward being dimensionless, the system is only informed how good or bad its current command is at following the target, but is not informed of the precise direction to choose to improve it. Its improvement will rely on “trials and errors”.

Considering the simpler the better, the reward is based on the subjective direction from the fovea $d(t)$:

$$r(t) = \frac{1}{2} - \left\| \overrightarrow{d(t)} \right\|$$

The rewards are expected to be negative most of the time at the beginning of the learning process, since the probability to have the target near the center of the fovea is initially low.

4.4 Learning

The learning process is applied for 250 s on both excitatory and inhibitory synapses, a reward is sent every τ_z ms (in figure 2, $\tau_z = 100$ ms). The learning parameter is $\alpha = 0.03/N^{(1)}$. The neural controller initially sends erratic commands to the effectors, which contributes to an approximately uniform exploration of various eye positions, while the target rapidly moves on the retina. The rewards being sent every 100 ms, a progressive improvement of the command is observable after 20-40 s where the target appears more frequently near the center of the retina (as the expectation of the reward increases). Then the behavior of the controller progressively stabilizes around 50-100 s and the mean distance remains stable (0.5-0.7 rad) for the rest of the simulation. The eye is now tracking the target, approximately reproducing the circular movement observed.

If we look at the premotor neurons responses individually at the end of the learning process, no specialization is clearly visible. Individual premotor neurons do not systematically respond to targets appearing on the opposite direction (not shown). If we now look at the output command (i.e. the mean of the 32 premotor neurons response), then a clear specialization is visible at the end of learning. In figure 2-C-, we give the mean motor response according to 256 typical target directions, at the beginning and at the end of the learning process. The initial movement field displays no particular organization, as it only relies on the randomness of the initial connections. On the contrary, the movements produced at the end of learning are systematically oriented toward the center of the retina, the command being weaker when the target is near the center, almost proportionally to the observed “error”.

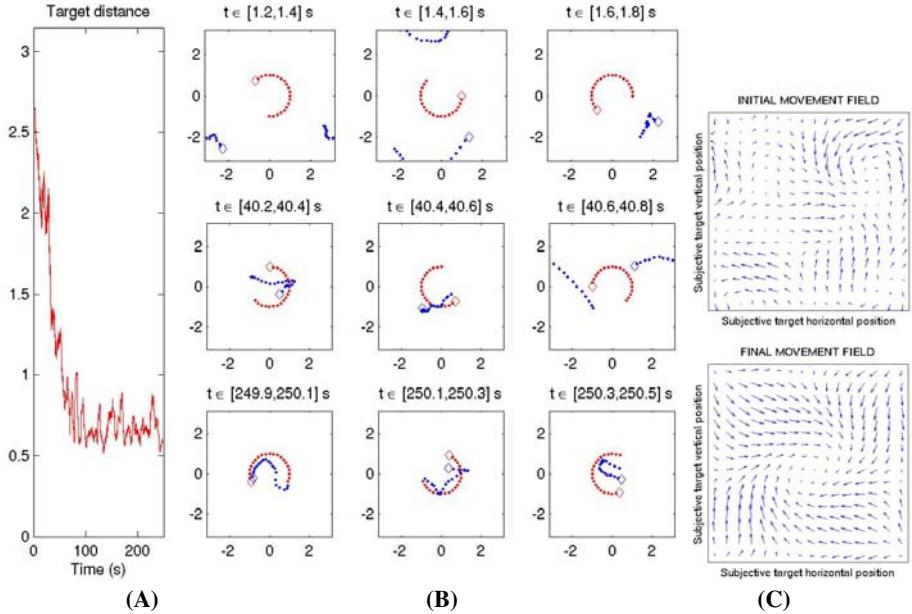


Fig. 2. (A) Evolution of the mean distance ($\|\psi - \theta\|$) on a 1 s interval, during the 250 s of the learning process. (B) Samples from the controller and target dynamics, on 200 ms intervals, with 10 ms resolution. The red dots give the target direction (the diamond gives the final direction in the interval). The blue dots give the eye direction. (C) (top) Initial average motor output for a target appearing at the considered subjective direction, (bottom) final average motor output (see text for more details).

We consider now the two parameters : τ_z and τ_{noise} . The value of τ_z controls both the length of the trace and the frequency of the rewards, the rewards appearing every τ_z ms and the trace taking into account the most recent firings in the same interval. We consider different values of τ_z from continuous rewards ($\tau_z = \delta t$) to sparse rewards ($\tau_z = 100$ ms). We observe in figure 3-A- that changing τ_z has no particular effect on the learning performance. The frequency of rewards is not found to significantly influence the learning process: the performance is about the same for rewards frequency within a [0.5...100] ms range.

The value of τ_{noise} controls the “slowness” of the noise (its autocorrelation) without modifying its variance. The case of $\tau_{noise} = \tau_m = 10$ ms roughly corresponds to a diffusive noise filtered by the neuron membrane, while the value $\tau_{noise} = \delta t$ corresponds to the simple gaussian escape noise model [6]. Changing the characteristics of the noise appears on the contrary to strongly impair the learning performance. With a simple uncorrelated gaussian noise ($\tau_{noise} = \delta t$), the controller is simply unable to learn the task. The “slowness” of the noise thus appears to be an essential feature for the achievement of learning. This effect can be explained by the closer correspondence between the noise and environment time constants. In particular, for the environment we consider, the autocorrelation window of the current error signal $\|\overrightarrow{\psi(t)} - \overrightarrow{\theta(t)}\|$ is of

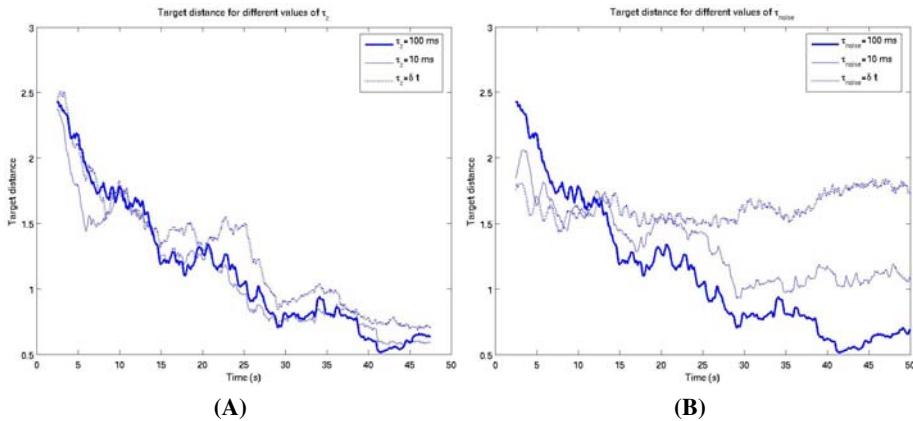


Fig. 3. (A) Mean distance during the first 50 s of the learning process, for $\tau_z = 100 \text{ ms}$ (bold line), $\tau_z = 10 \text{ ms}$ (thin line) and $\tau_z = \delta t$ (dotted line), with $\tau_{noise} = 100 \text{ ms}$. (B) Idem, for $\tau_{noise} = 100 \text{ ms}$ (bold line), $\tau_{noise} = 10 \text{ ms}$ (thin line) and $\tau_{noise} = \delta t$, with $\tau_z = 100 \text{ ms}$.

the order of 100 ms (not shown), which corresponds to the value of τ_{noise} giving the best learning performance.

5 Conclusions

We have presented a new learning setup that illustrates the relevance of a policy-gradient based RL approach for modelling realistic and convergent synaptic plasticity rules. We show here in particular that the shape of the noise is of crucial importance for the algorithm convergence. The time constants shared by the noise and the environment help the controller to capture the relevant shifts from the expected firing rates, and modify the weights accordingly, even when the reward information is sparse. During the learning process, the premotor neurons modify their response in order to favour better rewards, without getting finely specialized. It is only by looking at the population that a coherent response pattern appears, as a combination of individual outputs.

The realism of our approach in modelling natural learning depends on the existence of internal random signals having comparable self-correlation characteristics. Our modelling hypothesis should be considered regarding the known characteristics of spiking activity [11]. This study more generally points out the interest of temporally (and spatially) structured noises for orienting the “exploration” of the environment in artificial control systems.

Acknowledgements

The author thanks the INRIA Lille-Nord Europe for 1-year delegation in the SEQUEL team. This work is supported by the french ANR MAPS (ANR-07-BLAN-0335-02).

References

1. Watkins, C.J., Dayan, P.: Q-learning. *Machine learning* 8, 279–292 (1992)
2. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 229–256 (1992)
3. Bartlett, P.L., Baxter, J.: Hebbian synaptic modifications in spiking neurons that learn. Technical report, Research School of Information Sciences and Engineering, Australian National University (1999)
4. Sebastian Seung, H.: Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron* 40, 1063–1073 (2003)
5. Baras, D., Meir, R.: Reinforcement learning, spike time dependent plasticity and the bcm rule. *Neural Computation* 19(8), 2245–2279 (2007)
6. Gerstner, W., Kistler, W.: Spiking Neuron Models. Single Neurons, Populations, Plasticity. Cambridge University Press, Cambridge (2002)
7. Florian, R.V.: A reinforcement learning algorithm for spiking neural networks. In: Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), pp. 299–306 (2005)
8. Soula, H., Beslon, G., Mazet, O.: Spontaneous dynamics of asymmetric random recurrent spiking neural networks. *Neural Computation* 18, 60–79 (2006)
9. Sutton, R.S.: Learning to predict by the method of temporal differences. *Machine learning* 3, 9–44 (1988)
10. Hebb, D.: The Organization of behavior. Wiley, New York (1949)
11. Softky, W., Koch, C.: The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsps. *J. of Neuroscience* 13(1), 334–450 (1993)

How Bursts Shape the STDP Curve in the Presence/Absence of GABAergic Inhibition

Vassilis Cutsuridis^{1,*}, Stuart Cobb², and Bruce P. Graham¹

¹ Department of Computing Science and Mathematics, University of Stirling,
Stirling, FK9 4LA, UK

{vcu,b.graham}@cs.stir.ac.uk

² Division of Neuroscience and Biomedical Systems, University of Glasgow, Glasgow,
G12 8QQ, UK
sc107y@udcf.gla.ac.uk

Abstract. It has been known for some time that the synapses of the CA1 pyramidal cells are surprisingly unreliable at signalling the arrival of single spikes to the postsynaptic neuron [2]. On the other hand, bursts of spikes are reliably signalled, because transmitter release is facilitated. In the hippocampus, a single burst can produce long-term synaptic modifications. Bursts of spikes in addition to increasing reliability of synaptic transmission [3], they have been shown to provide effective mechanisms for selective communication between neurons in a network [4]. We investigate via computer simulations how the profile of spike-timing-dependent plasticity (STDP) in the CA1 pyramidal cell synapses is affected when an excitatory burst of spikes applied to dendrites is paired with an excitatory single spike applied to the soma in the absence and presence of a 100Hz GABAergic inhibitory spike train applied to the dendrites. We report that the shape of the STDP curve strongly depends on the burst interspike interval in the presence/absence of GABA_A when a presynaptic burst and a postsynaptic spike are paired together.

Keywords: Hippocampus, CA1 pyramidal neuron, computer model, STDP, GABA, LTP, LTD, calcium.

1 Introduction

Hebb's law states a synapse is strengthened only if the pre- and postsynaptic neurons are activated simultaneously [1]. STDP is a refinement of Hebb's law, which states that the precise timing of presynaptic and postsynaptic action potentials is actually the one that determines the sign and magnitude of synaptic modifications [2]. Bi and Poo showed that the profile of the STDP curve in the in-vitro hippocampal network has an asymmetrical shape with the largest LTP/LTD value at +/-10ms, respectively [7].

A recent study reported that the shape of the STDP profile depends on the location on the stratum radiatum (SR) dendrite [5]. A symmetric STDP profile is observed in the proximal SR dendrite, whereas an asymmetric one is observed

* Corresponding author.

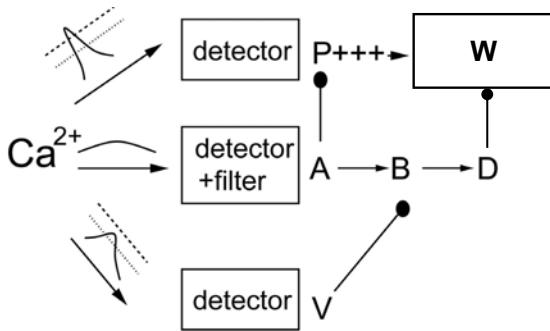


Fig. 1. Model calcium detection system [8], [11]. P detector: potentiation detector; D detector: depression detector; V detector: veto detector; W: synaptic weight.

in the distal one [6]. A symmetric STDP profile with short temporal window may serve as a coincidence detector between the incoming input and the output of the CA1 pyramidal cell, whereas an asymmetric profile with a broad temporal window may play a role in chunking of items in sequence learning. Recent computational works from our group predicted that the switching between operational modes (asymmetry-to-symmetry) is strongly dependent on the frequency band (theta vs. gamma) of the GABA_A inhibition, the conductance value of GABA_A inhibition and the relative timing between the GABAergic spike train and the pre- and post-synaptic excitation [8], [10]. A long-term potentiation (LTP) peak and two distinct long-term depression (LTD) tails of the symmetrical STDP curve were shown to be centered at +10 ms, +40 ms and -10 ms, respectively [8], [10]. The largest LTP value and the two distinct LTD tails were inversely proportional to the increase of GABA_A conductance [8], [10].

In this study we continue to investigate the asymmetry-to-symmetry transition in the CA1-SR synapses in the presence of complex inputs, such as bursts. We examine how the STDP profile in the CA1-SR pyramidal cell synapse is affected when a burst of excitatory spikes applied to the SR synapses is paired with a single excitatory spike applied to the soma in the absence and presence of a 100Hz GABAergic inhibitory spike train applied to the SR dendrites as a function of GABA_A conductance and burst interspike interval.

2 The Model

We used a Ca²⁺ dynamics model for the CA1 pyramidal cell [8], [11]. The model neuron had two compartments: a soma and a dendrite. The generation of action potentials was due to the interplay of a wealth of Na⁺, K⁺, Ca²⁺-activated K⁺ and Ca²⁺ currents as well as synaptic currents (AMPA and NMDA) [8], [9], [10]. Two excitatory transient inputs to the soma and SR dendrite were used to simulate the experimental STDP protocol. The mechanism for plasticity had a modular structure consisting of three biochemical detectors, which responded to the instantaneous calcium level in the SR dendrite. The detection system (see

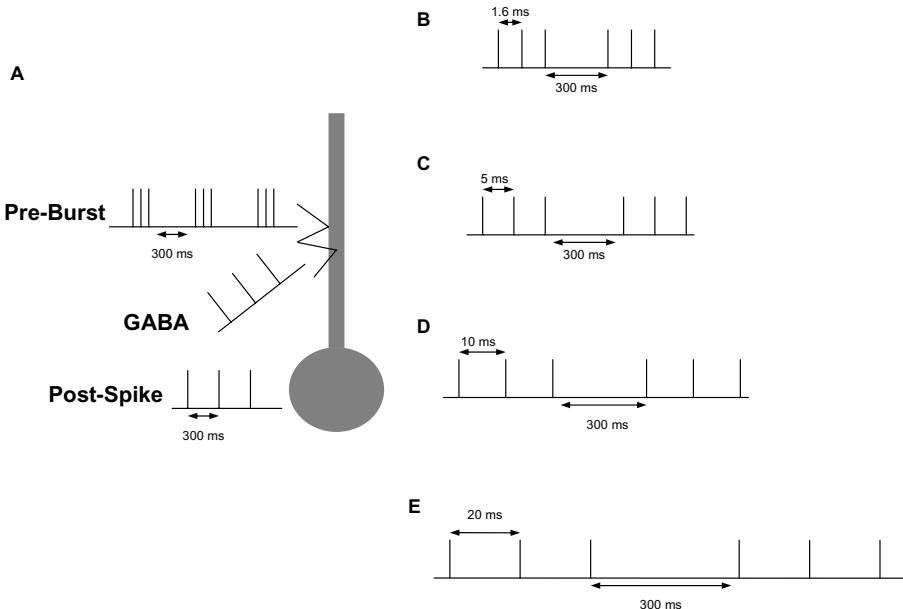


Fig. 2. (A) Our model CA1 neuron with its three transient inputs to the soma and SR dendrite. In all experimental paradigms, pairing takes place between an excitatory burst of spikes with T ms burst interspike interval, where T is a free parameter and an excitatory single spike (not shown) in the absence and presence of a 100Hz GABAergic spike train applied between the excitatory pair interval (not shown). (B) Experimental paradigm 1: burst interspike interval, T , is 1.6 ms (C) Experimental paradigm 2: burst interspike interval, T , is 5 ms (D) Experimental paradigm 3: burst interspike interval, T , is 10 ms (E) Experimental paradigm 4: burst interspike interval, T , is 20 ms.

fig. 1) consisted of: (1) a potentiation detector which detected calcium levels above a high-threshold (e.g. 4 M) and triggered LTP, (2) a depression detector which detected calcium levels exceeding a low threshold level (e.g. 0.6 M) and remained above it for a minimum time period and triggered LTD, and (3) a veto detector which detected levels exceeding a mid-level threshold (e.g. 2 M) and triggered a veto of the model's depression components. More details on the Ca^{2+} detectors system can be found in [8] and [10].

In this study we investigate how the pairing of a repeating excitatory burst applied to the SR dendrite and a repeating excitatory single spike applied to the soma affect the STDP curve in the presence and/or absence of a 100 Hz GABA_A inhibitory transient input to the SR dendrite.

3 Experiments

To investigate how the STDP curve is affected by the pairing of excitatory bursts and excitatory single spikes in the presence/absence of GABAergic inhibition,

we designed the following experimental protocol: Excitatory burst of spikes with T ms burst interspike interval and single spikes, which were repeatedly applied to the SR dendrite and soma, respectively, for 2 s (7 times at about 3 Hz) were paired in the absence and presence of a 100 Hz GABAergic inhibitory spike train applied between the excitatory pair interval $\Delta\tau$. Based on this protocol, we designed the following four physiological experiments (see figure 2), where the burst interspike interval, T, was allowed to vary:

- T = 1.6 ms
- T = 5 ms
- T = 10 ms
- T = 20 ms

During all experimental paradigms, we varied the conductance of $GABA_A$ and observed its effects on the amplitude of the proximal SR Ca^{2+} spike and the STDP curve. These results are reported in the next section.

4 Results

4.1 Pairing of an SR Burst and a Somatic Spike in the Absence of $GABA_A$ as a Function of Burst Interspike Interval

Figure 3 depicts the saturated synaptic weight values (W_∞) as a function of the interstimulus interval, $\Delta\tau = t_{post} - t_{pre}$. $\Delta\tau$ is the interstimulus interval between the first presynaptic spike of the burst (a presynaptic burst is composed of three spikes with varying interspike interval) and the postsynaptic spike. Simulations were performed with $\Delta\tau$ ranging from -100 to 100 in increments of 10 ms. When the burst interspike interval is 1.6 ms, a symmetric STDP profile is evident. The largest LTP value is at 10 ms. The duration of the temporal window for learning is 30 ms, where beyond that only forgetting (i.e LTD) is taking place. When the burst interspike interval is increased to 5 ms, an asymmetric STDP profile appears. The largest LTP and LTD values are at 20 ms and -10 ms, respectively. At 10 ms burst interspike interval the asymmetric STDP profile is maintained and all W_∞ values are positive and larger than 0.1. This means that only LTP (i.e. learning) is possible when a CA3 Schaffer collateral burst and a CA1 output spike are paired in the absence of GABA, but how strong learning is depends on $\Delta\tau$. As 20 ms burst interspike interval, another symmetric curve appears with the largest LTP value at 60 ms.

4.2 Pairing of an SR Burst and a Somatic Spike in the Presence of a 100Hz GABA Spike Train in the $\Delta\tau$ Interval

Figure 4 is a composite figure of four graphs of W_∞ vs $\Delta\tau$ as a function of burst interspike interval and GABA conductance. It is clear from all four graphs that at GABA presence and as the conductance of GABA increases a symmetric

STDP profile appears. Particularly interesting is the case of 20 ms burst interspike interval, where when GABA is present, even at low conductance values, "catastrophic forgetting" (i.e. LTD for all $\Delta\tau$ values) is present.

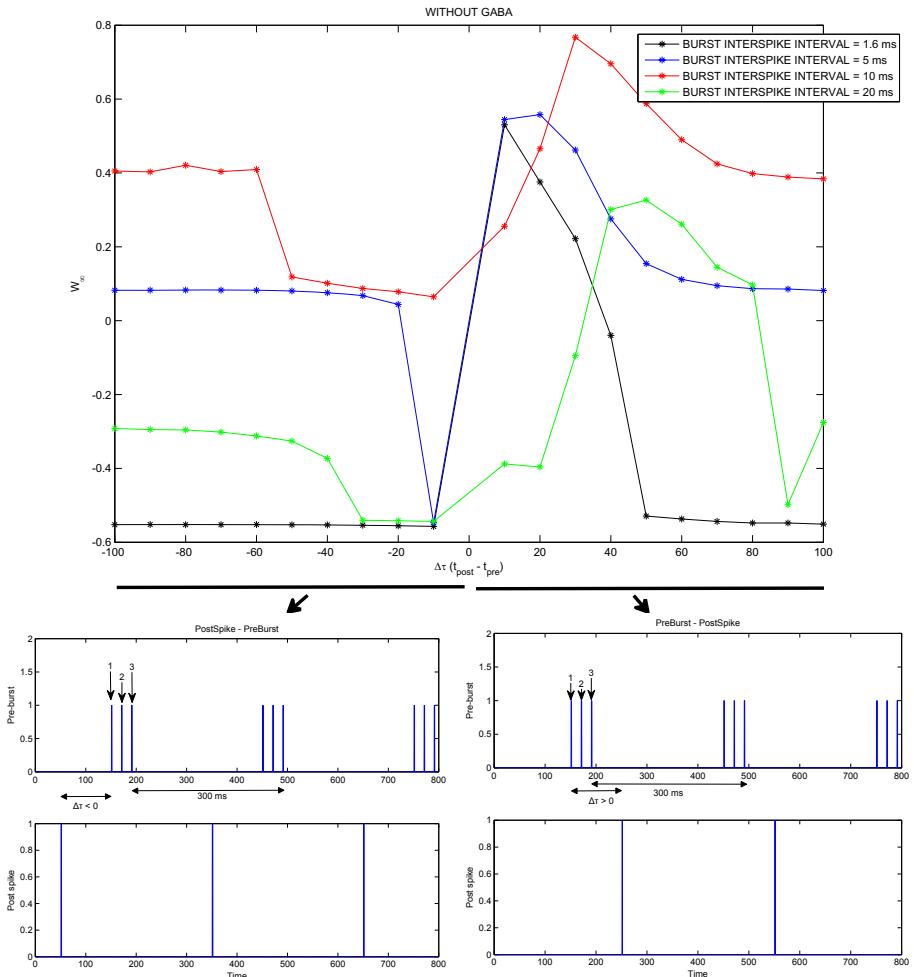


Fig. 3. (Top) Simulated asymmetric STDP profile as a function of burst interspike intervals in the absence of $GABA_A$. $\Delta\tau (t_{post} - t_{pre})$ is the interstimulus interval between the first presynaptic spike of the burst and the postsynaptic spike. $\Delta\tau$ ranges from -100 to 100 in increments of 10 ms. Solid lines with arrows point to the bottom two figures, which depict the relative timing between the presynaptic burst and the postsynaptic spike. (Bottom-left) PostSpike - PreBurst scenario, where postsynaptic single spike precedes the presynaptic burst, comprised of three spikes, by $\Delta\tau$. $\Delta\tau$ takes values from -10 ms to -100 ms. The pairing repeats every 300 ms. (Bottom-right) PreBurst - PostSpike scenario, where a presynaptic burst precedes the postsynaptic spike by $\Delta\tau$. $\Delta\tau$ takes values from +10 ms to +100 ms. The pairing repeats every 300 ms.

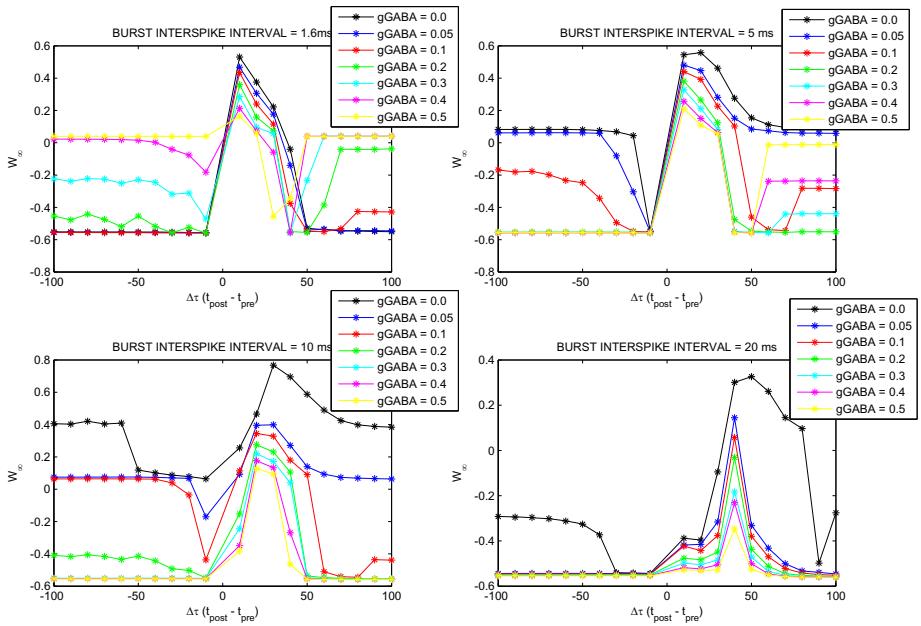


Fig. 4. STDP profiles from the pairing of a burst of spikes with T ms burst interspike interval, where T is a free parameter, applied to SR synapses and a single spike applied to the soma in the absence and presence of a 100 Hz GABA spike train as a function of $GABA_A$ conductance. $\Delta\tau$ ($t_{post} - t_{pre}$) ranges from -100 to 100 in increments of 10ms. (Top-left) Burst interspike interval, T , is 1.6 ms. (Bottom-left) Burst interspike interval, T , is 5 ms. (Top-right) Burst interspike interval, T , is 10 ms. (Bottom-right) Burst interspike interval, T , is 20 ms.

Figure 5 is a composite figure of the W_∞ vs $\Delta\tau$ graph as a function of a 5 ms burst interspike interval and increasing $GABA_A$ conductance ($g_{GABA_A} = 0.0$ mS/cm² and $g_{GABA_A} = 0.1$ mS/cm²) and the $[Ca^{2+}]_P$ (i.e. potentiation) and LTD agents (V, A, B, and D) vs time graphs for B, C, D, E, F, G, H, and I. W_∞ values of the W_∞ vs $\Delta\tau$ graph. In the paired protocol of a presynaptic burst stimulation followed by a postsynaptic single spike stimulation of 10 ms later (preBurst-10-postSpike) in the absence of $GABA_A$, we see a large influx of calcium through the NMDA channel due to removal of the magnesium block by the back propagating action potential (BPAP) (top figure 5B). In the preBurst-50-postSpike scenario, the influx of calcium is reduced because the arrival of the BPAP at the dendrite comes later. While the slow closing of the NMDA channels still allow calcium influx to be enhanced by this unblocking, the peak calcium level is lower in the preBurst-40-postSpike scenario than in the preBurst-10-postSpike (top figure 5C) because NMDA channels are more inactivated by the time the magnesium is removed. This leads to a large potentiation (i.e. P) spike (medium figure 5B) and small LTD agents' profiles (bottom figure 5B)

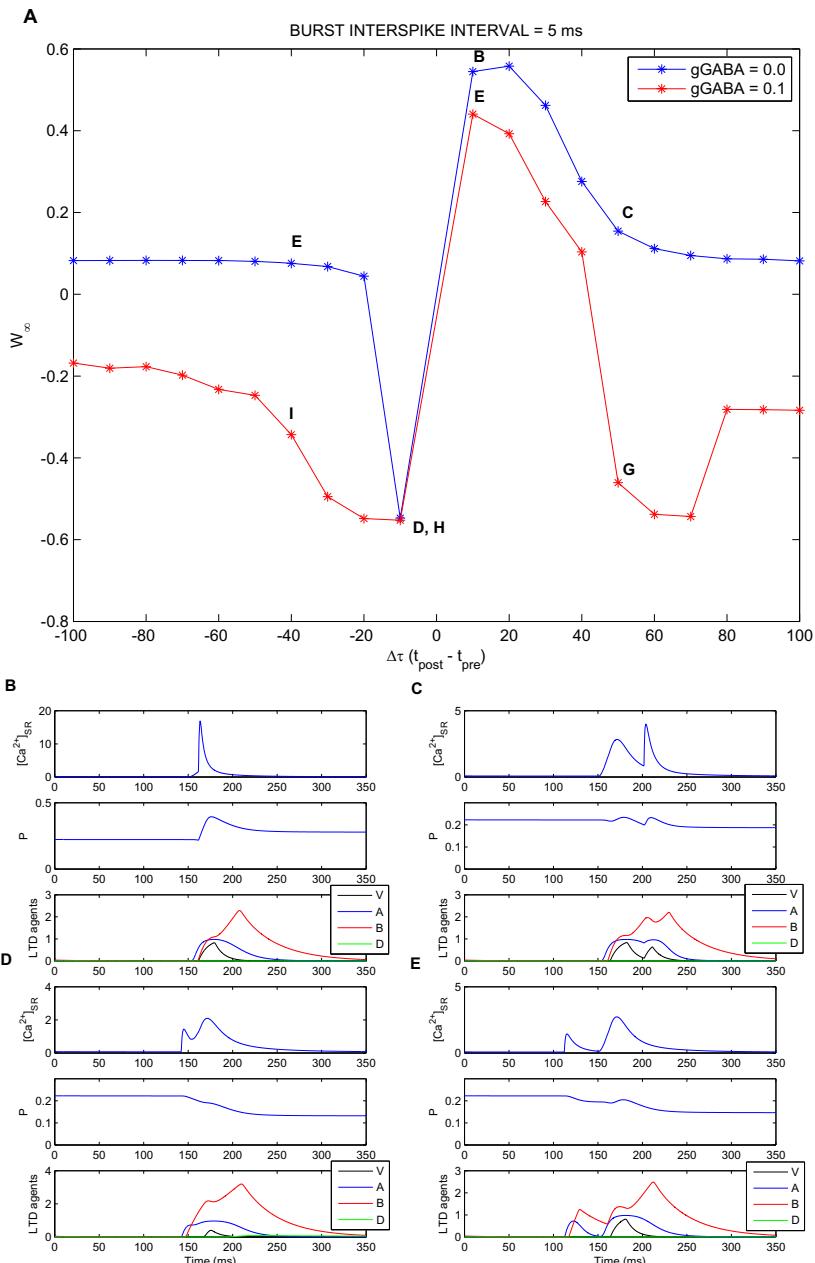
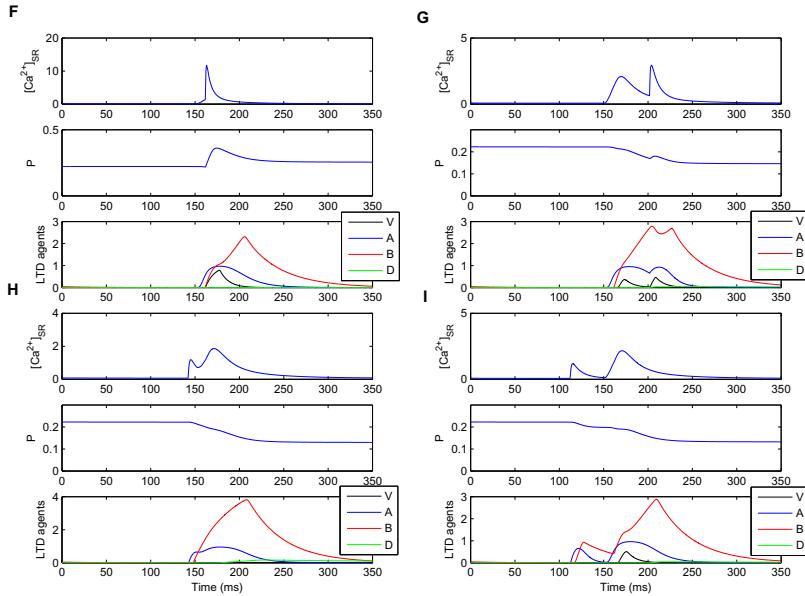


Fig. 5. (A) STDP profile from the pairing of a burst of spikes with 5 ms burst interspike interval applied to SR synapses and a single spike applied to the soma in the absence and presence of a 100 Hz GABA_A spike train as a function of GABA_A conductance. $\Delta\tau$ (t_{post} - t_{pre}) ranges from -100 to 100 in increments of 10 ms. (B-I) Composite figures corresponding to points B through I in figure 5A of $[\text{Ca}^{2+}]$, P (potentiation) and LTD agents (V, A, B, and D) with respect to time.

**Fig. 5. (continued)**

when compared with preBurst-50-postSpike case (medium and bottom figure 5C). The peak calcium level will continue to decrease as the preBurst-postSpike interstimulus interval is lengthened.

When $GABA_A$ inhibition is present the peak calcium level in the preBurst-10-postSpike (top figure 5F) is lower than in preBurst-10-postSpike case in the absence of $GABA_A$ (top figure 5B). In the preBurst-50-postSpike in the presence of $GABA_A$ scenario (top figure 5G), the effect of $GABA_A$ on W_∞ is more pronounced than in preBurst-10-postSpike without $GABA_A$ case. This is due to the increased number of $GABA_A$ spikes in the pre-post intersimulus interval (6 spikes in the 50 ms pre-post interstimulus interval vs 2 spikes in the 10 ms pre-post interstimulus interval).

In the postSpike-before-preBurst synaptic pairing with a 10 ms interval (postSpike-10-preBurst), a much smaller calcium enters the dendrite through the L-type voltage gated calcium channels (VGCCs) (top figure 5D). The effect of $GABA_A$ on calcium influx through the VGCCs is negligible and hence the LTD value of the W_∞ in the presence of $GABA_A$ (top figure 5H) is the same as in the absence of it (top figure 5D). The potentiation curves (medium figures 5D and 5H) are tonically decreasing. From the LTD agents (bottom figures 5D and 5H), the veto signal is low and hence its effect on the B curve is small. The tall B curve leads to a depression spike (i.e. D variable) and hence to LTD.

In the postSpike-40-preBurst without $GABA_A$ scenario, the calcium influx (top figure 5E) is greater than in the postSpike-10-preBurst (top figure 5D). Hence, a small but pronounced P spike is evident (medium figure 5E). A larger veto spike prevents the depression curve (D variable) from affecting the W_∞

value. In the postSpike-40-preBurst with GABA_A scenario, the calcium influx (top figure 5I) is slightly greater than in the postSpike-10-preBurst (top figure 5H). Hence, a very small P spike is evident (medium figure 5I). A veto spike is present but not sufficient strong to prevent the depression curve (D variable) from affecting the W_∞ , hence resulting to LTD.

5 Conclusion

A Ca^{2+} dynamics model of the CA1 pyramidal neuron with three calcium amplitude detectors was used to study the effects of GABAergic interneurons to the symmetry-to-asymmetry transition of the STDP profile in the proximal SR dendrite. In support of previous computational work from our group [8, 10], which predicted that the symmetry-to-asymmetry transition is strongly dependent on the frequency band (theta vs. gamma), the conductance value of GABA_A inhibition and the relative timing between the GABAergic spike train and the pre-post interstimulus interval, the symmetry-to-asymmetry-back-to-symmetry transition is also dependent on the burst interspike interval in the presence/absence of GABA_A when a presynaptic burst and a postsynaptic single spike are paired. In the future, we intend to investigate this transition even further for other pre-post burst and single spike pairings in the presence of different GABA_A gamma frequency sub-bands and conductance values.

References

- [1] Hebb, D.O.: *The organization of behavior*. John Wiley, New York (1949)
- [2] Allen, C., Stevens, C.F.: An evaluation of causes for unreliability of synaptic transmission. *Proc. Natl. Acad. Sci. U.S.A.* 91(22), 10380–10383 (1994)
- [3] Lisman, J.: Bursts as a unit of neural information: making unreliable synapses reliable. *TINS* 20(1), 38–43 (1997)
- [4] Izhikevich, E.M., Desai, N.S., Walcott, E.C., Hoppensteadt, F.C.: Bursts as a unit of neural information: selective communication via resonance. *TINS* 26(3), 161–167 (2003)
- [5] Nishiyama, M., Hong, K., Mikoshiba, K., Poo, M., Kato, K.: Calcium stores regulate the polarity and input specificity of synaptic modification. *Nature* 408, 584–589 (2000)
- [6] Tsukada, M., Aihara, T., Kobayashi, Y., Shimazaki, H.: Spatial Analysis of Spike-Timing-Dependent LTP and LTD in the CA1 Area of Hippocampal Slices Using Optical Imaging. *Hippocampus* 15, 104–109 (2005)
- [7] Bi, G.Q., Poo, M.M.: Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength and postsynaptic cell type. *J. Neurosci.* 18, 10464–10472 (1998)
- [8] Cutsuridis, V., Cobb, S., Graham, B.P.: A Ca^{2+} dynamics model of the STDP symmetry-to-asymmetry transition in the CA1 pyramidal cell of the hippocampus. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) *ICANN 2008, Part II. LNCS*, vol. 5164, pp. 627–635. Springer, Heidelberg (2008)

- [9] Cutsuridis, V., Cobb, S., Graham, B.P.: A CA1 Heteroassociative Microcircuit Model of the Hippocampus. In: Proc. of the AREADNE: Research in Encoding and Decoding of Neural Ensembles, Santorini, Greece, June 26-29 (2008)
- [10] Cutsuridis, V., Cobb, S., Graham, B.P.: Modelling the STDP Symmetry-to-Asymmetry Transition in the Presence of GABAergic Inhibition. Neural Network World (in press)
- [11] Rubin, J.E., Gerkin, R.C., Bi, G.Q., Chow, C.C.: Calcium time course as signal for spike-timing-dependent plasticity. *J. Neurophysiol.* 93, 2600–2613 (2005)

Optimizing Generic Neural Microcircuits through Reward Modulated STDP

Prashant Joshi* and Jochen Triesch

Frankfurt Institute for Advanced Studies,
Ruth-Moufang Str. 1, 60438, Frankfurt, Germany
[{joshi, triesch}@fias.uni-frankfurt.de](mailto:{joshi,triesch}@fias.uni-frankfurt.de)
<http://www.fias.uni-frankfurt.de/~{joshi, triesch}>

Abstract. How can we characterize if a given neural circuit is optimal for the class of computational operations that it has to perform on a certain input distribution? We show that modifying the efficacies of recurrent synapses in a generic neural microcircuit via spike timing dependent plasticity (STDP) can optimize the circuit in an unsupervised fashion for a particular input distribution if STDP is modulated by a global reward signal. More precisely, optimizing microcircuits through reward modulated STDP leads to a lower eigen-value spread of the cross-correlation matrix, higher entropy, highly decorrelated neural activity, and tunes the circuit dynamics to a regime that requires a large number of principal components for representing the information contained in the liquid state as compared to randomly drawn microcircuits. Another set of results show that such optimization brings the mean firing rate into a realistic regime, while increasing the sparseness and the information content of the network. We also show that the performance of optimized circuits improves for several linear and non-linear tasks.

Keywords: Reward modulated STDP, generic neural microcircuits, optimization.

1 Introduction

Reservoir computing paradigms, such as the Liquid State Machine (LSM) and the Echo State Network (ESN) have been proposed as a plausible mechanism through which neural readouts can perform diverse computational tasks in parallel [123]. Although computationally similar, LSMs have been typically discussed in context of computations carried out by generic cortical microcircuits composed of spiking neurons, while ESNs have been conventionally used to provide high performance on a number of engineering tasks.

* We acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) within the "Bernstein Focus: Neurotechnology" through research grant 01GQ0840, the European Union through project MEXT-CT-2006-042484 and the Hertie Foundation. We would also like to thank Herbert Jaeger, Ueli Rutishauser and Răzvan Florian for helpful discussions.

A key feature of existing reservoir computing models is that learning is constrained to only the synapses projecting from the generic neural microcircuit to the linear readout, leaving the recurrent circuitry intact. While this kind of learning is sufficient for a large repertoire of computational tasks, an obvious next step is to optimize the neural circuit for the class of computational operations that the circuit has to perform on a particular input distribution [4, 5, 6]. Determining whether a particular neural circuit is optimal for the computational tasks that it has to perform is not an easy question as the relationship between dynamical properties of the input and output streams and the properties of circuit dynamics induced through inputs is not well understood [4].

It has been proposed that a neural circuit that has been optimized for a certain input distribution will demonstrate the following four properties: a low eigenvalue spread¹ of the cross-correlation matrix of the liquid state $\mathbf{x}(t)$ ², a higher value of liquid-state entropy³, highly decorrelated neural activity, and a large number of principal components needed to represent the information contained in the liquid state as compared to a random microcircuit [4]. Loosely speaking, all these four measures aim at making the individual reservoir units as mutually different as possible.

This article demonstrates that a neural circuit can be optimized in the above sense for a particular input distribution via STDP, if the STDP is being modulated by a global reward signal. This optimization is done in an unsupervised fashion. Results indicate that the approach presented here is quite robust and brings the circuit dynamics into a more biologically realistic regime of lower firing rate, high degree of sparseness and increases the information content per spike. Further results indicate that such an optimized circuit increases the performance of linear readouts on a set of benchmark computational tasks.

2 Methods

The experiments described in this article were performed on generic cortical microcircuit models composed of leaky integrate-and-fire neurons⁴. These circuits

¹ Eigen-value spread is defined as the ratio $|\lambda_{max}/\lambda_{min}|$, where λ_{max} and λ_{min} are the maximum and minimum eigen values of the cross-correlation matrix of liquid state.

² The liquid state $\mathbf{x}(t)$, is obtained by applying a low-pass filter to the spike trains emitted by the neurons in the generic neural microcircuit model. More precisely, each component of $\mathbf{x}(t)$ models the impact that a particular neuron v may have on the membrane potential of a generic readout neuron. Thus each spike of neuron v is replaced by a pulse whose amplitude decays exponentially with a time constant of 30 ms.

³ Is defined as $-\sum_{i=1}^n x_i(t) \cdot \ln(x_i(t))$, where $x_i(t)$ is the i^{th} component of the liquid state $\mathbf{x}(t)$, and n is the number of neurons in the generic neural microcircuit.

⁴ Neuron Parameters: Membrane time constant 30 ms, absolute refractory period 3 ms (excitatory neurons), 2 ms (inhibitory neurons), threshold 15 mV (for a resting membrane potential assumed to be 0), reset potential drawn uniformly from the interval [13.8 mV, 14.5 mV].

were not created for any specific computational task. Sparse synaptic connectivity between neurons was generated with a biologically realistic bias towards short-range connections by a probabilistic rule, and synaptic parameters were chosen randomly from distributions that depended on the type of pre- and postsynaptic neurons in accordance with empirical data from [78]. The neurons in the generic neural microcircuit models were placed on the integer-points of a 3-D grid, and 20% of these neurons were randomly chosen to be inhibitory. The probability of a synaptic connection from neuron a to neuron b (as well as that of a synaptic connection from neuron b to neuron a) was defined as $C \cdot \exp(-D^2(a, b)/\lambda^2)$, where $D(a, b)$ is the Euclidean distance between neurons a and b , and λ (set to 2) is a parameter which controls both the average number of connections and the average distance between neurons that are synaptically connected. Depending on whether the pre- and postsynaptic neurons were excitatory (E) or inhibitory (I), the value of C was set according to [7] to 0.3 (EE), 0.2 (EI), 0.4 (IE), 0.1 (II).

The generic microcircuit models that were optimized and used for various computational tasks described in this article were similar in structure to those that were earlier used for various computational tasks in open loop [91011] and closed-loop [121314] setups. Results described in this article were obtained using a generic neural microcircuit consisting of 135 neurons arranged on the grid points of a $3 \times 3 \times 15$ cube in 3D. Inputs to the circuit were 16 Poisson spike trains drawn independently at 40 Hz, with each input spike train being projected to approximately 63% of randomly chosen neurons in the circuit.

Initially, the circuit was simulated once for 200 ms to measure the baseline values of the eigen-value spread (ε), entropy (H), pair-wise correlation (ρ), and number of principal components (N) needed to represent 95% of the information contained in the liquid state. Each subsequent trial lasted for 200 ms and consisted of simulating the circuit with inputs drawn from the distribution and measuring the global reward signal at the end of the trial. The reward for the k^{th} trial was computed as:

$$r = \alpha \cdot (-\Delta\varepsilon + \Delta H - \Delta\rho + \Delta N) \quad (1)$$

where $\Delta(\cdot)$ denotes the percentage difference in the value of a parameter between k^{th} and $(k-1)^{th}$ trial and α is a scaling factor that was set to 2.5.

The recurrent synapses were modified using STDP which was modulated using the reward signal mentioned above. More precisely, for a presynaptic spike occurring at time t_{pre} , and a postsynaptic spike occurring at time t_{post} , the weight change Δw , of the synapse was given by:

$$\Delta w = \begin{cases} r \cdot A_+ \cdot e^{-\frac{t_{post}-t_{pre}}{\tau_+}} & , \text{ if } t_{post} - t_{pre} > 0 \\ r \cdot A_- \cdot e^{\frac{t_{post}-t_{pre}}{\tau_-}} & , \text{ if } t_{post} - t_{pre} < 0 \end{cases} \quad (2)$$

where A_+ (A_-) define the peaks and τ_+ (τ_-) reflect the time constants of the exponential STDP kernel⁵ for potentiation (depression). It should be noted that

⁵ STDP kernel parameters: $A_+ = 3e^{-10}$, $A_- = -3e^{-10}$, $\tau_+ = 40$ ms, and $\tau_- = 50$ ms.

the reward modulated STDP as described by equation 2 does not change the inherent nature of the synapse, i.e. the paradigm does not convert an excitatory synapse into an inhibitory one, or vice-versa.

3 Results

Can the learning rule (see eqns. 1 and 2) optimize a neural circuit and what impact does such optimization has on various aspects of circuit dynamics? This was studied through an experiment which consisted of simulating the network for 20 runs, each consisting of 10 trials while reward modulated STDP was used to change the recurrent synaptic weights.

Figure 1A-C present the results from a single run of 10 trials. Fig. 1 demonstrates the impact of optimizing circuit dynamics through reward modulated STDP. Panel A shows the input to the circuit (16 Poisson spike trains drawn independently at 40 Hz). Panels B and C show respectively, the response of the circuit to this stimulus before and after optimization, while the corresponding change in mean firing rate (population average) of the network can be seen in Panel D. It is interesting to note that the optimization process results in bringing the firing activity of neurons to a more realistic, sparse regime.

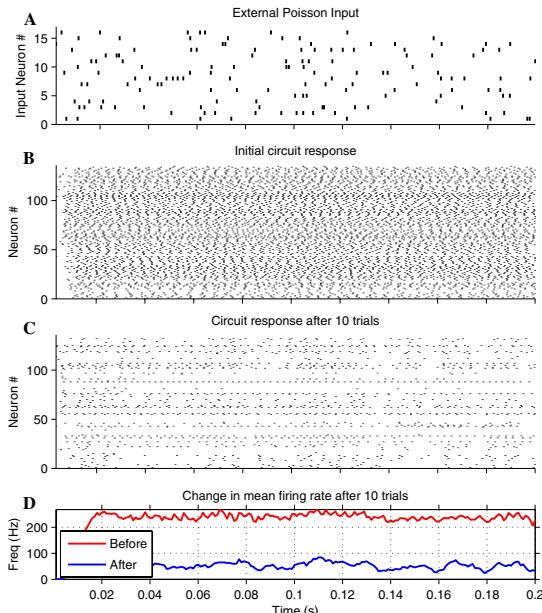


Fig. 1. (A) The external Poisson input spike trains to the neural microcircuit. Each of the input spike trains was projected to approximately 63% of neurons in the circuit. Response of the circuit (B) before and (C) after optimization using reward modulated STDP for 10 trials. (D) The change in mean firing rate. The optimization process results in bringing the firing activity of neurons to a more realistic regime.

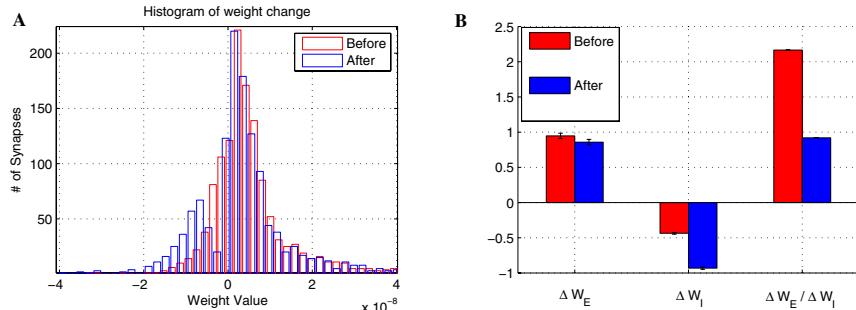


Fig. 2. Change in weights through reward modulated STDP after 10 trials. (A) Histograms showing the distribution of weights before and after optimization. (B) The change in excitatory synapses (ΔW_E), inhibitory synapses (ΔW_I), and the change in the ratio of excitation to inhibition ($\Delta W_E/\Delta W_I$) before (red) and after (blue) optimization. The error measures are standard error (SE), and the values of ΔW_E and ΔW_I have been scaled by 10^8 .

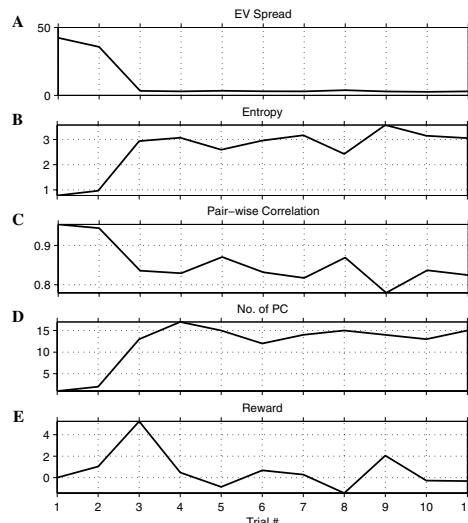


Fig. 3. Evolution of the (A) eigen-value spread, (B) entropy, (C) pair-wise correlation, and (D) the number of principal components needed to represent 95% of information contained in the liquid state. Panel (E) shows the value of the corresponding global reward signal during the 10 trials.

Figure 2 shows the change in recurrent synaptic weights occurring as a result of using reward modulated STDP for 10 trials. Panel A shows the histogram of weight values before and after optimization. It can be observed that the circuit

response becomes more sparse after optimization (see Fig. II) as the learning rule increases the global inhibition as seen by larger values for inhibitory weights. Panel B shows the change in weights of excitatory synapses (ΔW_E), inhibitory synapses (ΔW_I), and the ratio of excitation to inhibition ($\Delta W_E/\Delta W_I$). Note that the optimization process brings the circuit into a regime of balanced excitation and inhibition.

Figure 3 shows the evolution of 4 parameters and the corresponding reward signal for this experiment. Note that the eigen-value spread and pair-wise correlation⁶ decrease while the entropy and the number of principal components increase over the course of experiment as desired.

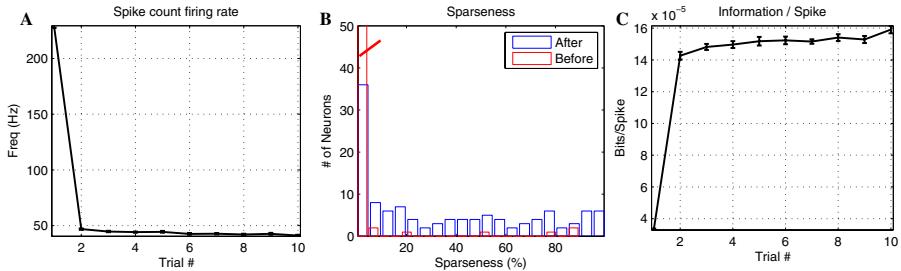


Fig. 4. Evolution of (A) mean firing rate, (B) neuronal sparseness and (C) the information content per spike in the generic neural circuit. Results shown in the figure are means with SE measures computed over 20 runs, each run consisting of 10 trials. In panel B, the bar indicating the neurons in the circuit before optimization that had 0% sparseness has been truncated (slanted red line).

Figure 4 presents the result that demonstrate the effect of reward modulated STDP on the mean firing rate, the sparseness of firing, and the average information content of the network. As seen in panel A of the figure, the mean firing rate of the network went down as a result of applying the learning rule. At the same time, applying the learning rule increased the sparseness of neural activity. We used the same measure of sparseness as in [15,16]. More precisely, the sparseness S , of a neuron was defined as:

$$S = \frac{1 - [(\sum r_i/K)^2 / \sum(r_i^2/K)]}{1 - (1/K)} \quad (3)$$

where $K = 20$ is the total number of runs and r_i is the response of the neuron to the i^{th} stimulus (averaged across runs). Each stimulus was a set of 16 independent Poisson spike trains drawn at 40 Hz. Values of S near 0% indicate a dense code, and values close to 100% indicate a sparse code. Panel B shows the

⁶ A decrease in pair-wise correlation implies that action potentials of individual neurons in the circuit get more decorrelated.

histogram of sparseness of neurons in the circuit before and after optimization. There is a significant increase in neuronal sparseness as a result of optimization.

The average information content of the network also increased progressively with the optimization process as seen in panel C of Fig. 4. More precisely, the amount of information about the inputs to the circuit contained per single spike in the circuit activity increased with trials. In other words, application of the learning rule brings the circuit dynamics to a regime with low redundancy, where each single spike in the network carries more meaning and information.

Optimization from the Perspective of Linear Readouts

A key hypothesis of optimizing generic neural microcircuits for a particular input distribution is that it will improve the performance of linear readouts attached to the circuit which are assigned different computational tasks. This was demonstrated through another experiment whose results are shown in Fig. 5. Four readouts were trained using simple linear regression, to perform different linear

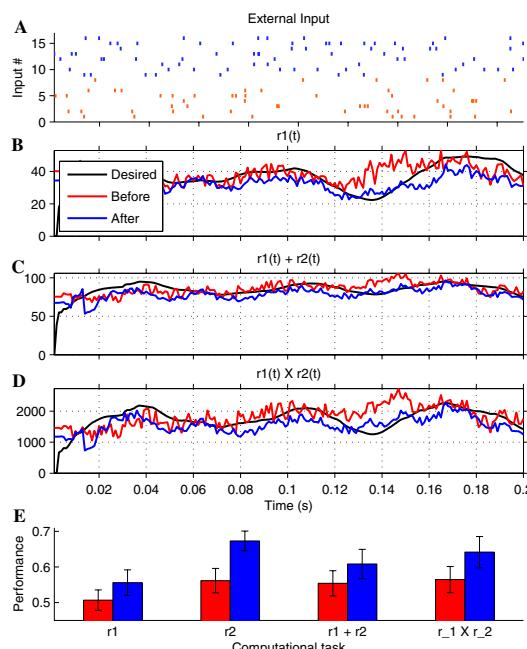


Fig. 5. (A) 8 external Poisson spike trains corresponding to each of the rates $r_1(t)$ and $r_2(t)$ (B) The target (black), and the observed performance of readouts connected to optimized (blue) and unoptimized (red) neural circuits, which were computing the value of $r_1(t)$. Similar plots for readouts computing (C) $r_1(t) + r_2(t)$, and (D) $r_1(t) \times r_2(t)$. (E) The readouts show significant increase in performance on each of the tasks, when the circuits have been optimized (blue bars) compared to unoptimized circuits (red bars).

and non-linear computational tasks on the incoming Poisson spike trains. The performance of these readouts was compared for the case when an unoptimized generic neural microcircuit was being used to the case when the circuit had been optimized for 10 trials. Inputs to the circuit were 16 Poisson spike trains drawn independently at 40 Hz (like other experiments). The task of the first readout was to compute the instantaneous firing rate $r_1(t)$ of the first 8 input channels. The second readout was computing $r_2(t)$, the rate for the next 8 input channels. The third readout was computing the sum of rates $r_1(t) + r_2(t)$, while the fourth readout was performing the non-linear computation of product of rates, $r_1(t) \times r_2(t)$. After 100 training trials, the performance of readouts was subsequently validated for another 50 validation trials, using inputs that weren't seen during training. Panel A shows the 8 external Poisson spike trains corresponding to each of the rates $r_1(t)$ and $r_2(t)$. Panel B shows the target (black), and the observed performance of readouts connected to optimized (blue) and unoptimized (red) circuits. Panel C and D show similar results for readouts computing

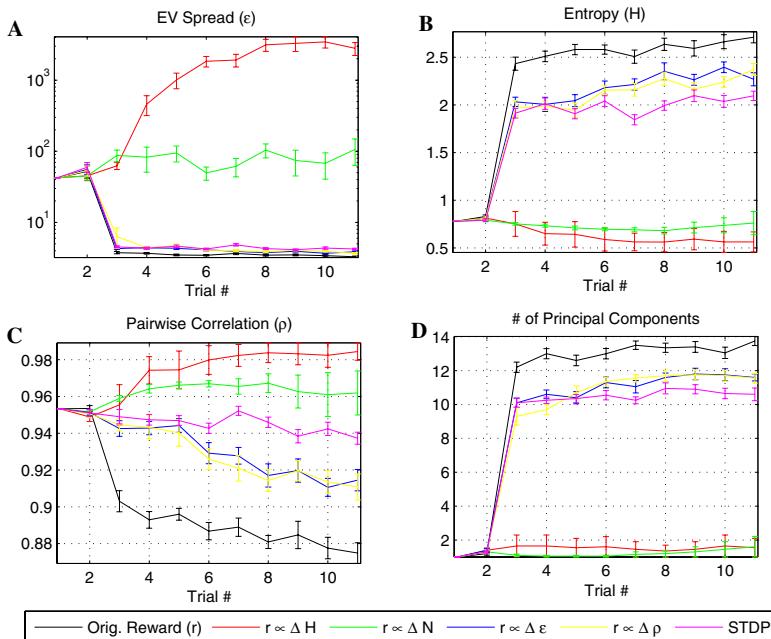


Fig. 6. Comparison of different reward functions. For each reward function, 20 runs were performed, each consisting of 10 trials. Also shown in each panel are comparison with the control case when the synapses were modified using STDP only. Shown are results for evolution of (A) EV Spread, (B) Entropy, (C) Pair-wise correlation, and (D) Number of principal components needed to represent 95% of information in the liquid state. Note that the y-axis of panel A uses logarithmic scale.

$r1(t) + r2(t)$ and $r1(t) \times r2(t)$. Each of the 4 readouts showed an increase in performance⁷ when they were seeing dynamics of a circuit that had been optimized through reward modulated STDP as can be seen in panel D. The errorbars are standard error measure.

Other Possible Reward Functions

Can the global reward be computed by other reward functions besides the one described by eqn. II? This question was investigated through another experiment that recorded how the 4 measures, i.e. ε , H , ρ and N evolved over 20 runs, each composed of 10 trials, when we used 5 different reward functions. Besides the reward function described earlier (eqn. II), each of the other 4 reward functions were computed as:

$$r = \pm \hat{\alpha} \cdot \Delta(.) \quad (4)$$

where $\hat{\alpha}$ is a scaling factor set to 10, and $\Delta(.)$ is the percentage change in one of the four parameters, ε , H , ρ and N . The sign of reward depended on which of the 4 parameters was chosen to construct the reward function. It can be seen from Fig. 6 that the original reward function gives the best performance, reward functions based on ε and ρ work slightly better than the control case, i.e. when the synapses were modified using STDP only, and reward functions based on H and N did not work at all.

4 Discussion

This article presents an unsupervised learning rule to modify the recurrent synaptic efficacies of a generic neural microcircuit via reward modulated STDP, such that the circuit is optimized for a set of computational tasks that it has to perform on a certain input distribution. Optimizing neural circuits with the learning rule described here leads to a lower eigen-value spread of the cross correlation matrix of the liquid state, a higher value of liquid state entropy, highly decorrelated neural activity, and brings the circuit dynamics to a regime where a large number of principal components are required to represent the information contained in liquid state.

Results indicate that optimizing the neural circuits in the manner described here leads to a regime of balanced excitation and inhibition, realistic firing rates, higher sparseness and increases the information content per spike in the network. It should be noted that such sparse code is computationally efficient and at the same time biologically realistic. For example, it has been observed that V1 neurons of macaques encode information using a sparse code [15].

Linear readouts connected to circuits that have been optimized for a particular input distribution perform significantly better on the same computational tasks as compared to readouts that see dynamics from a randomly drawn generic neural circuit. Thus, from the perspective of a linear readout, the learning rule

⁷ Performance was measured as the correlation between the target and observed signals.

described here brings the circuit dynamics to a regime where it becomes easier for the readouts to do several different linear and non-linear computational tasks.

Another interesting result was the possibility to compute the global reward based on how either of the four measures evolved. It was shown that the original reward function (eqn. II) performed better than other possible functions. The functions based on ε and ρ performed slightly better than the control case (using STDP without reward), and functions based on H and N did not work.

References

1. Jaeger, H., Maass, W., Principe, J.: Special issue on Echo State Networks and Liquid State Machines. *Neural Networks* 20, 287–289 (2007)
2. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14(11), 2531–2560 (2002)
3. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* 304, 78–80 (2004)
4. Jaeger, H.: Reservoir riddles: Suggestions for Echo State Network research. In: IJCNN, pp. 1460–1462 (2005)
5. Lazar, A., Pipa, G., Triesch, J.: Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Networks* 20, 312–322 (2007)
6. Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J., Stroobandt, D.: Improving reservoirs using intrinsic plasticity. *Neurocomputing* 71, 1159–1171 (2008)
7. Gupta, A., Wang, Y., Markram, H.: Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science* 287, 273–278 (2000)
8. Markram, H., Wang, Y., Tsodyks, M.: Differential signaling via the same axon of neocortical pyramidal neurons. *PNAS* 95, 5323–5328 (1998)
9. Buonomano, D.V., Merzenich, M.M.: Temporal information transformed into a spatial code by a neural network with realistic properties. *Science* 267, 1028–1030 (1995)
10. Maass, W., Markram, H.: On the computational power of recurrent circuits of spiking neurons. *Journal of Computer and System Sciences* 69(4), 593–616 (2004)
11. Maass, W., Natschläger, T., Markram, H.: Computational models for generic cortical microcircuits. Chapman & Hall/CRC, Boca Raton (2004)
12. Maass, W., Joshi, P., Sontag, E.D.: Computational aspects of feedback in neural circuits. *PLOS Computational Biology* 3(1), e165, 1–20 (2007)
13. Joshi, P., Maass, W.: Movement generation with circuits of spiking neurons. *Neural Computation* 17(8), 1715–1738 (2005)
14. Joshi, P.: From memory based decisions to decision based movements: A model of interval discrimination followed by action selection. *Neural Networks* 20, 298–311 (2007)
15. Vinje, W.E., Gallant, J.L.: Sparse coding and decorrelation in primary visual cortex during natural vision. *Science* 287, 1273–1276 (2000)
16. Billimoria, C.P., Kraus, B.J., Narayan, R., Maddox, R.K., Sen, K.: Invariance and sensitivity to intensity in neural discrimination of natural sounds. *Jour. of Neurosc.* 28(25), 6304–6308 (2008)

Calcium Responses Model in Striatum Dependent on Timed Input Sources

Takashi Nakano^{1,2}, Junichiro Yoshimoto^{1,2}, Jeff Wickens², and Kenji Doya^{1,2}

¹ Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan

² Initial Research Project, Okinawa Institute of Science and Technology
12-22 Suzaki, Uruma, Okinawa 904-2234, Japan
{nakano,jun-y,wickens,doya}@oist.jp

Abstract. The striatum is the input nucleus of the basal ganglia and is thought to be involved in reinforcement learning. The striatum receives glutamate input from the cortex, which carries sensory information, and dopamine input from the substantia nigra, which carries reward information. Dopamine-dependent plasticity of cortico-striatal synapses is supposed to play a critical role in reinforcement learning. Recently, a number of labs reported contradictory results of its dependence on the timing of cortical inputs and spike output. To clarify the mechanisms behind spike timing-dependent plasticity of striatal synapses, we investigated spike timing-dependence of intracellular calcium concentration by constructing a striatal neuron model with realistic morphology. Our simulation predicted that the calcium transient will be maximal when cortical spike input and dopamine input precede the postsynaptic spike. The gain of the calcium transient is enhanced during the “up-state” of striatal cells and depends critically on NMDA receptor currents.

1 Introduction

The striatum is the input site of the basal ganglia and receives glutamate inputs from the cerebral cortex and thalamus and dopamine inputs from the substantia nigra. Both the glutamatergic and dopaminergic afferents terminate on the dendritic spines of medium spiny neurons, the output neurons of the striatum. Plasticity of the cortico-striatal synapses depends on both dopamine input and increase in the intracellular calcium concentration, a form of dopamine-dependent and calcium-dependent plasticity [1,2].

Recently, two groups studied spike timing-dependent plasticity (STDP) of cortico-striatal synapses and reported contradictory results: One reported that cortical input repeatedly preceding the postsynaptic spike induced LTP whereas cortical input following the postsynaptic spike induced LTD [3] ; but the other reported the opposite coupling, this is, which cortical input following the postsynaptic spike induced LTD and cortical input preceding the postsynaptic spike induced LTP [4]. Although the literature [3] suggested that the activation of D1 receptors was necessary to induce both LTD and LTP, the effect of timing of dopamine input on STDP was still open.

It has been shown in cortical and hippocampal neurons that there is a threshold level of influx of calcium ions to the postsynaptic spines which determines whether a synapse is potentiated or depressed [5][6][7][8]. Synaptic efficacy is unaffected with no calcium influx, depressed by an intermediate level of calcium elevations, and potentiated with a large increase in calcium levels. We previously constructed a molecular cascade model in the striatal medium spiny neuron and predicted that the calcium influx as well as dopamine receptor activation would strongly affect bidirectional plasticity of cortico-striatal synapses [2]. Integrating these findings, the dynamics of the calcium influx is likely to be an important factor to uncover the mechanisms behind STDP in cortico-striatal synapses.

In this study, we constructed a multi-compartmental model of the striatal neuron with realistic morphology and examined how the timing of cortical input, dopamine input, and postsynaptic spiking affect calcium influx.

2 Methods

2.1 Electrophysiological Experiment

Acute corticostriatal slices ($300\text{ }\mu\text{m}$ thickness) were prepared from p21-25 DRD1a EGFP Swiss Webster mice [9]. Whole cell recordings were made in current clamp mode from identified GFP-positive cells, which express D1 dopamine receptors. Cell identification was achieved using an Olympus BX51WI microscope and data were acquired using an Axon Multiclamp 700B amplifier and pClamp 10 software.

To obtain morphological images, a neuron was filled with biocytin through the patch pipette and was tagged with Alexa 488 after the whole cell recording.

Morphological Modeling. Our model originated from Wolf's study [10]. Based on our measurements of actual medium spiny neurons, we extended Wolf's model to include dendrites. This allowed us to precisely evaluate the effect of backpropagating action potentials, which could not be accurately modelled in Wold's

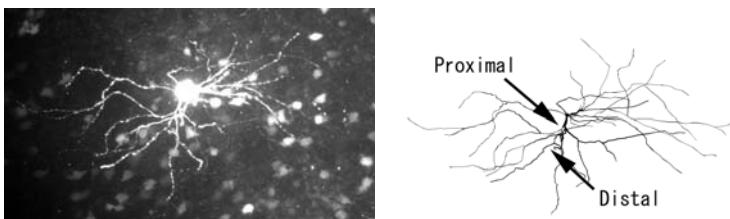


Fig. 1. The morphology of a D1-type dopamine receptor-expressing medium spiny neuron. Left: a medium spiny neuron was filled with the fluorescent dye Alexa 488 and observed using a DSU confocal microscope. Right: morphological data imported into the NEURON simulator. The arrows point indicate the proximal and distal dendrites described in Section 3, respectively.

deduced model. The 3D morphological image (Fig. 1) was obtained using Neurolucida neuronal tracing system with a DSU confocal microscope. Cell morphology was manually traced using Neurolucida. The traced data include information about the lengths and diameters of the dendrites [11].

Spatial discretization of model neurons was done automatically according to the d_lambda rule [11]. We adjusted the values of membrane resistance R_m and the membrane capacitance C_m using the standard methods [12][13]. We added two spines with diameter of 1 μm and length of 1.2 μm at proximal and distal dendrites 25 μm and 50 μm away from the soma, respectively, in order to measure the calcium transient in the different spines (see arrows in Fig. 1).

2.2 Ionic Currents

Our model had the same ionic channels as Wolf's [10]. There were two types of sodium channels, fast (NaF) and persistent (NaP), and six different potassium channels, inwardly rectifying (KIR), slow A-type (KAs), fast A-type (KAf), 4-AP resistant persistent (KRP), small conductance calcium dependent (SK), and large-conductance calcium dependent (BK), and six calcium currents: N-, Q-, R-, T-, Cav1.2 [high-voltage activated (HVA)] L- and Cav1.3 [low-voltage activated (LVA)] types [14]. The model also had ionic currents through AMPA- and NMDA-type glutamate receptors and GABA receptors.

We adjusted the conductance parameters to fit our experimental data using Neurofitter [15].

Calcium Dynamics. The original model [10] included four processes contributing to calcium dynamics, namely calcium influx from AMPA and NMDA-type synaptic receptors, voltage-gated calcium channels, calcium buffer, and calcium pumps. In addition, we added calcium release from the intracellular calcium store (endoplasmic reticulum, ER), through ryanodine and IP3 (inositol-1,4,5-triphosphate) channels [16].

The transient change in the intracellular calcium $[Ca]_i$ was modeled as

$$\frac{d[Ca]_i}{dt} = k(J_{CICR} + J_{IP_3} - J_{Uptake} + J_{Leak} + J_{Channel} + J_{Synaptic} - J_{Pump}) + ([Ca]_0 - [Ca]_i)/\tau_r. \quad (1)$$

Here, J_{CICR} is the flux caused by calcium-induced calcium release from intracellular stores. This process is mediated by a ryanodine receptor; thus, we modeled it as

$$J_{CICR} = V_{CICR} \frac{[Ca]_i}{[Ca]_i + K_{CICR}} ([Ca]_{ER} - [Ca]_i).$$

J_{IP_3} is the flux caused by IP3-induced calcium release from intracellular stores. $[Ca]_{ER}$ is the calcium concentration in ER. It is known that the process has a bell-shaped steady state curve dependent on $[Ca]_i$ with a sharp peak around 0.2 uM; thus, we modeled it as

$$J_{IP_3} = V_{IP_3} m^3 h^3 ([Ca]_{ER} - [Ca]_i),$$

where an activation gate m and an inactivation gate h ,

$$m = \frac{[IP_3]_i}{[IP_3]_i + d_{IP_3}} \frac{[Ca]_i}{[Ca]_i + d_{act}},$$

$$h = \frac{d_{inh}([IP_3]_i + d_{IP_3})}{d_{inh}([IP_3]_i + d_{IP_3}) + [Ca]_i([IP_3]_i + d_{dis})}.$$

IP3 (inositol-1,4,5-triphosphate) was generated via G-proteins when glutamate binds to mGluRs. The transient change in the level of intracellular IP3, $[IP_3]$ was modeled by

$$\frac{d[IP_3]_i}{dt} = \gamma t \exp(-t/t_{peak}) - \beta([IP_3]_i - [IP_3]_{min}).$$

J_{Uptake} is the calcium update to the endoplasmic reticulum (ER), which was modeled as

$$J_{\text{Uptake}} = V_{\text{Uptake}} \frac{[Ca]_i^2}{K_{\text{Uptake}}^2 + [Ca]_i^2}.$$

J_{leak} is the calcium leak from the ER, which was modeled as

$$J_{\text{Leak}} = V_{\text{Leak}} ([Ca]_{ER} - [Ca]_i).$$

J_{Channel} and J_{Synaptic} are calcium influx through calcium channels and calcium permeable glutamate receptors respectively. J_{Pump} is pumping activity to outside the cell and last term in equation 1 is simple diffusive or buffering process.

Dopamine Modulation. To reflect a finding that dopamine D1 agonists increase inward K currents and L-type calcium currents [17], we introduced a dopamine modulation factor μ , which scaled Kir and L-type calcium currents. μ gradually increased from $\mu = 1$ (corresponding to the basal dopamine level) and reached a peak ($\mu \simeq 2$) at 50 milliseconds after the dopamine signal arrived [18].

3 Results

We calibrated the model parameters to fit the electrophysiological properties of medium spiny neurons expressing D1-type dopamine receptors, which were examined by whole cell patch clamp experiments in vitro. Fig. 2 compares the current pulse responses of the model and those of a real neuron. The model replicated the characteristic properties of the medium spiny neurons: resting membrane potential around -85mV, small voltage responses to hyperpolarizing current, and shallow after-hyperpolarization (AHP) after spike firing.

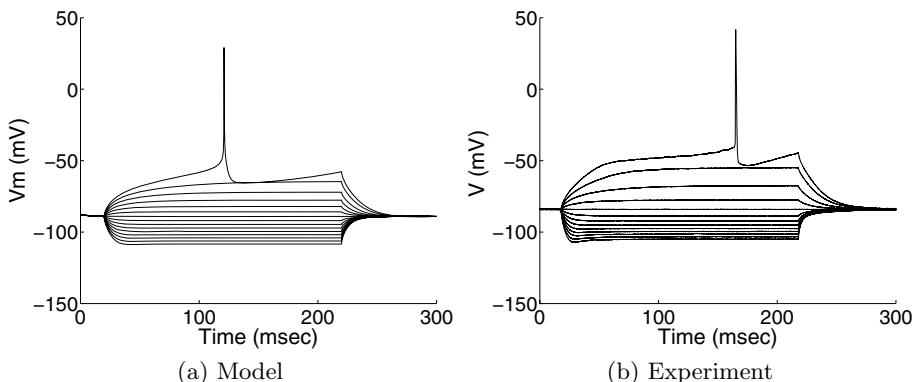


Fig. 2. Model and experimental responses of medium spiny neurons to current steps from -0.3 nA to 0.24 nA at intervals of 0.04 nA

Using this model, we investigated the following: 1) the voltage and calcium responses of the dendrites to glutamate input and back-propagation action potentials; 2) the timing dependence of calcium response to presynaptic inputs and postsynaptic spike; 3) the triplet interactions of glutamate input, dopamine input, and postsynaptic spike. We compared these properties at proximal and distal dendrites under different membrane potential levels, known as up-state and down-state [19].

3.1 Voltage and Calcium Responses

We investigated voltage and calcium responses in dendritic spines on proximal and distal dendrites. The membrane potential of medium spiny neurons with intact cortical input fluctuates between the “down-state” at about -85 mV and the “up-state” at about -50 mV [19]. We simulated the up-state by a steady current input of 0.2 nA to the soma and the down-state by no current input.

Fig. 3 (a-d) shows the responses of the spine in a proximal dendrite to glutamatergic synaptic inputs at proximal spine. The EPSPs and calcium responses were facilitated in the up-state (Figs. 3(a) and (b)). The increased calcium influx in the up-state was mediated by voltage-gated calcium channels and NMDA receptors as well as calcium-dependent calcium release from the ER (compare Figs. 3(c) and (d)).

Fig. 3(e-h) shows the responses of the proximal spine to a postsynaptic spike caused by a transient current pulse (2.5 nA, 2 msec). Again, the EPSPs and calcium responses were enhanced in the up-state (Figs. 3(e) and (f)), but the calcium influx caused by the bAP was mainly from L-type calcium channels both in the down-state and the up-state (Figs. 3(g) and (h)). These properties held for the responses of the distal spine except that the magnitude of the bAP and the calcium flux were reduced compared to the case of the proximal spine (data not shown).

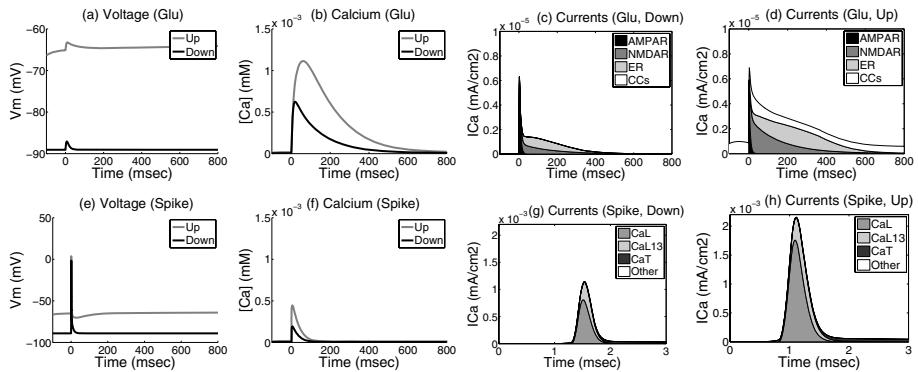


Fig. 3. Voltage and calcium responses of the proximal dendritic spine to glutamate input (corresponding to panels (a-d)) and somatic spike (corresponding to panels (e-f)). Panels (a,e) and (b,f) show time courses of voltage and intracellular calcium concentration, respectively, where the black lines indicate down-state and the gray lines indicate up-state simulated by a steady current input. Panels (c,g) and (d,h) shows the distribution of sources of calcium influx to the spine in the down-state and the up-state, respectively. In the legends, AMPAR, NMDAR, ER, CCs, CaL, Cal13 and CaT in the legends denotes calcium influx from AMPA receptors, NMDP receptors, ER, all calcium channels, L-type calcium channels (v1.2), L-type calcium channel (v1.3) and T-type calcium channel respectively.

3.2 Timing of Presynaptic Input and Postsynaptic Spike

Next, we studied how the neuronal responses depended on the relative time of paired inputs: 1) glutamate input and post-synaptic spike; and 2) dopamine input and post-synaptic spike. Fig. 4(a) shows the calcium responses to glutamate input (upper panel) and dopamine input (lower panel) preceding (gray) or succeeding (black) a postsynaptic action potential by 20 ms in down-state. For both glutamate and dopamine inputs, pre-post timing caused more calcium response than post-pre timing. Fig. 4(b) shows the pre-post timing dependence of peak calcium responses.

Fig. 4(c) shows the sources of calcium influx. For the glutamate input (upper panels), the increased calcium response in the pre-post protocol was due to enhanced calcium influx through NMDA receptors following bAP (upper panel). This is consistent with previous findings [20,21]. For dopamine input, the increased calcium response for pre-post protocol was due to enhanced calcium influx through L-type calcium channels (lower panel).

3.3 Triplet Interaction

We then investigated the dependence of calcium responses to the timing of both glutamate and dopamine inputs relative to the postsynaptic spike. Fig. 5 shows the peak calcium concentration in the proximal spine (a-c) and the distal spine

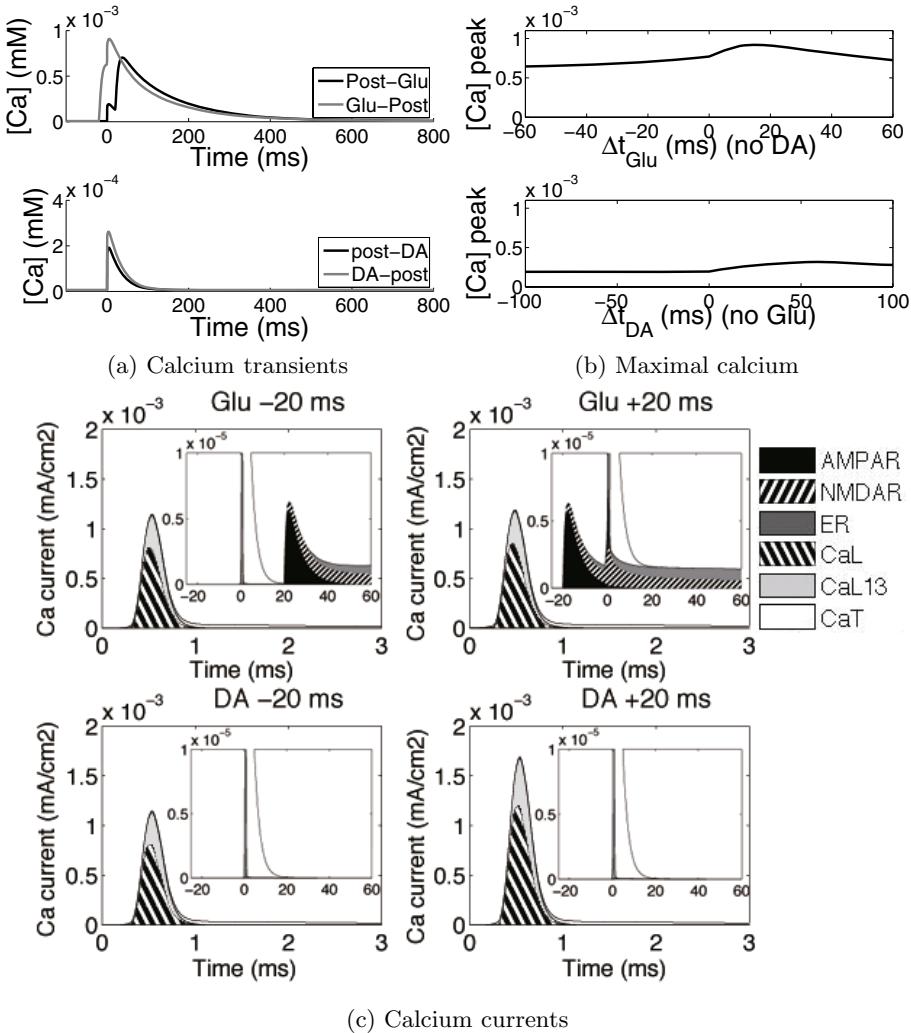


Fig. 4. The dependence of the calcium response on the timing of presynaptic input (glutamate or dopamine) relative to postsynaptic action potential in the down-state. (a) Calcium transients in case of presynaptic glutamate (corresponding to upper panel) and dopamine (corresponding to lower panel) inputs. The gray lines indicate presynaptic input leading the postsynaptic spike by 20 ms, and the black lines indicate the presynaptic input following the postsynaptic spike by 20 ms. (b) Peak of the calcium transient for each timed input. Δt_{Glu} (or Δt_{DA}) means the time difference from presynaptic glutamate (or dopamine) input to postsynaptic action potential. Positive time means the presynaptic input precedes the postsynaptic action potential. (c) Distribution of sources of calcium current in the proximal dendrite. The conditions are $\Delta t_{Glu} = -20$ (in the upper-left), $\Delta t_{Glu} = +20$ (in the upper-right), $\Delta t_{DA} = -20$ (in the lower-left), $\Delta t_{DA} = +20$ (in the lower-right). The insertion in the upper-right of each panel is the magnification. The abbreviations in the legend are same as those in Fig. 3.

(d-f). In the proximal spine in the down-state (a), the effect of dopamine input was most pronounced when the dopamine input preceded the postsynaptic spike by about 50 ms. On the other hand, at the distal spine (d), the modulation by DA input timing was not very significant. In the distal dendrite, the peak of calcium was lower than that in the proximal dendrite because the bAP was attenuated at the distal dendrite and such a small bAP was not enough to activate L-type calcium channels. Timing dependence of the glutamate and postsynaptic spike was weakened by blocking NMDA receptors. Timing dependence of the dopamine and postsynaptic spike disappeared when L-type calcium channel was blocked out in down-state though the result is left out here for lack of space. Therefore, the triplet condition can be regarded as generalization of two timed-input conditions shown in Section 3.2. However, it is noteworthy that the modulation by dopamine input was not just a linear summation. Calcium levels were elevated drastically only when the glutamate and dopamine inputs arrived at an adequate timing preceding post-synaptic input.

Fig. 5 (b) and (e) show that the calcium responses were enhanced in the up-state. Their peaks were shifted to $\Delta t_{\text{Glu}} = +50\text{ms}$ from $\Delta t_{\text{Glu}} = +15\text{ms}$ in the down-state. Interestingly, these timing-dependence almost disappeared when all NMDA receptors were blocked, as shown in Fig. 5 (c) and (f). This suggests that NMDA receptors are the major component regulating the timing-dependent calcium response.

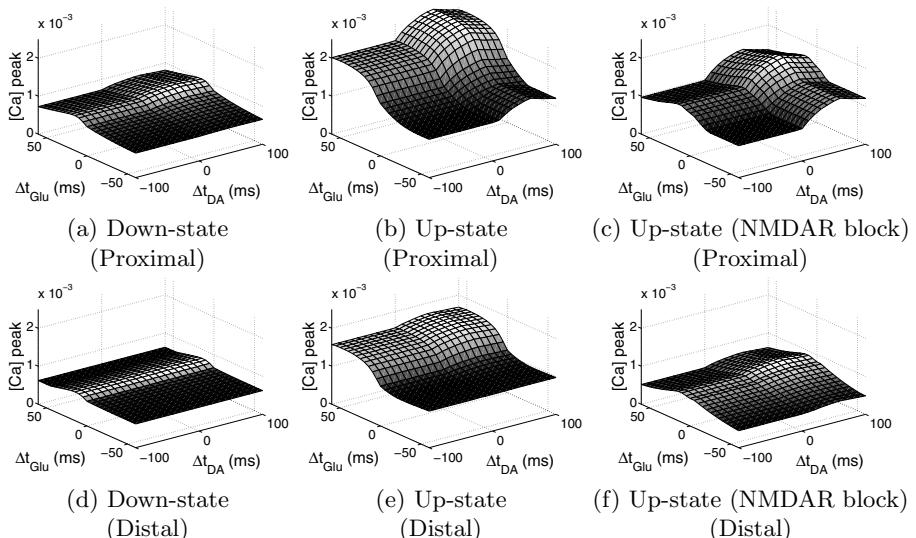


Fig. 5. Peak calcium in the proximal spine (a-c) and distal spine (d-f) in response to different timings of glutamate and dopamine inputs relative to a postsynaptic spike. The spine of panels (a,d) and (b,e) was in the down-state and the up-state, respectively. The spine of panels (c,f) was in the up-state but the NDMA receptors were blocked.

4 Conclusion and Discussion

We constructed a multi-compartment model of a medium spiny neuron of the striatum based on real morphological data. The major findings by simulation of the model were: 1) Glutamate input preceding postsynaptic spike increased calcium levels, mostly caused by NMDA receptors. 2) Dopamine input preceding the postsynaptic spike also increased calcium levels, due to modulation of the L-type calcium channels. 3) The dopamine timing effect was observed in proximal spines when both glutamate and dopamine inputs preceded the postsynaptic spike. 4) In the up-state, dopamine input preceding post-synaptic input elevated calcium levels, even in distal spines.

Assuming that moderate and intense calcium responses induce LTD and LTP, respectively, our simulation results support the report by [3] rather than that by [4]; thus, glutamate preceding postsynaptic spike would induce LTP and the opposite timed input would induce LTD.

However, we cannot absolutely deny the report in [4]. According to our supplementary simulations (not shown here), different durations of a post-synaptic pulse lead to different calcium transients; e.g. when the post-synaptic pulse has a longer duration (30 ms), peak calcium transients in the up-state were lower than those in the down-state for many combinations of timings of three input sources. Combined with the fact that [4] tested post-synaptic pulses with longer duration to generate a spike than [3], the results implied a possibility that the conflict between [3] and [4] might be caused by the details of the post-synaptic stimuli in the different protocols.

References

1. Reynolds, J., Wickens, J.: Dopamine-dependent plasticity of corticostriatal synapses. *Neural Netw.* 15(4-6), 507–521 (2002)
2. Nakano, T., Doi, T., Yoshimoto, J., Doya, K.: A kinetic model of dopamine and calcium dependent striatal synaptic plasticity (submitted, 2009)
3. Pawlak, V., Kerr, J.N.D.: Dopamine receptor activation is required for corticostriatal spike-timing-dependent plasticity. *J. Neurosci.* 28(10), 2435–2446 (2008)
4. Fino, E., Glowinski, J., Venance, L.: Bidirectional activity-dependent plasticity at corticostriatal synapses. *J. Neurosci.* 25(49), 11279–11287 (2005)
5. Lisman, J.: A mechanism for the hebb and the anti-hebb processes underlying learning and memory. *Proc. Natl. Acad. Sci. USA* 86(23), 9574–9578 (1989)
6. Artola, A., Singer, W.: Long-term depression of excitatory synaptic transmission and its relationship to long-term potentiation. *Trends in Neurosciences* 16(11), 480–487 (1993)
7. Hansel, C., Artola, A., Singer, W.: Relation between dendritic Ca^{2+} levels and the polarity of synaptic long-term modifications in rat visual cortex neurons. *Eur. J. Neurosci.* 9(11), 2309–2322 (1997)
8. Zucker, R.S.: Calcium- and activity-dependent synaptic plasticity. *Current Opinion in Neurobiology* 9(3), 305–313 (1999)

9. Gong, S., Zheng, C., Doughty, M.L., Losos, K., Didkovsky, N., Schambra, U.B., Nowak, N.J., Joyner, A., Leblanc, G., Hatten, M.E., Heintz, N.: A gene expression atlas of the central nervous system based on bacterial artificial chromosomes. *Nature* 425(6961), 917–925 (2003)
10. Wolf, J.A., Moyer, J.T., Lazarewicz, M.T., Contreras, D., Benoit-Marand, M., O'Donnell, P., Finkel, L.H.: NMDA/AMPA ratio impacts state transitions and entrainment to oscillations in a computational model of the nucleus accumbens medium spiny projection neuron. *Journal of Neuroscience* 25(40), 9080–9095 (2005)
11. Hines, M.L., Carnevale, N.T.: Neuron: a tool for neuroscientists. The Neuroscientist: a review journal bringing neurobiology, neurology and psychiatry 7(2), 123–135 (2001)
12. Segev, I.: Single neurone models: oversimple, complex and reduced. *Trends in Neurosciences* 15(11), 414–421 (1992)
13. Koch, C.: Biophysics of computation (January 2004)
14. Catterall, W.A.: Structure and regulation of voltage-gated Ca^{2+} channels. *Annu. Rev. Cell. Dev. Biol.* 16, 521–555 (2000)
15. Geit, W.V., Achard, P., Schutter, E.D.: Neurofitter: A parameter tuning package for a wide range of electrophysiological neuron models. *Frontiers in Neuroinformatics* (January 2007)
16. Koch, C., Segev, I.: Methods in neuronal modeling: From ions to networks, p. 671 (January 1998)
17. Surmeier, D.J., Bargas, J., Hemmings, H.C., Nairn, A.C., Greengard, P.: Modulation of calcium currents by a D1 dopaminergic protein kinase/phosphatase cascade in rat neostriatal neurons. *Neuron* 14(2), 385–397 (1995)
18. Gruber, A.J., Solla, S.A., Surmeier, D.J., Houk, J.C.: Modulation of striatal single units by expected reward: a spiny neuron model displaying dopamine-induced bistability. *J. Neurophysiol.* 90(2), 1095–1114 (2003)
19. Wilson, C.J., Kawaguchi, Y.: The origins of two-state spontaneous membrane potential fluctuations of neostriatal spiny neurons. *J. Neurosci.* 16(7), 2397–2410 (1996)
20. Carter, A.G., Sabatini, B.L.: State-dependent calcium signaling in dendritic spines of striatal medium spiny neurons. *Neuron* 44(3), 483–493 (2004)
21. Kerr, J.N.D., Plenz, D.: Action potential timing determines dendritic calcium during striatal up-states. *J. Neurosci.* 24(4), 877–885 (2004)

Independent Component Analysis Aided Diagnosis of Cuban Spino Cerebellar Ataxia 2

Rodolfo V. García¹, Fernando Rojas², Jesús González², Belén San Román³, Olga Valenzuela⁴, Alberto Prieto², Luis Velázquez⁵, and Roberto Rodríguez⁵

¹ Network Department, U. of Holguín, Cuba, Spanish MAEC-AECID fellowship

² Department of Computer Architecture and Technology, U. of Granada, Spain

³ PhD Student, University of Granada, Spain

⁴ Department of Applied Mathematics, University of Granada, Spain

⁵ Centre for the Research and Rehabilitation of Hereditary Ataxias “Carlos J. Finlay”, Holguín, Cuba

Abstract. Precedent studies have found abnormalities in the oculomotor system in patients with severe SCA2 form of autosomal dominant cerebellar ataxias (ADCA), including the latency, peak velocity, and deviation in saccadic movements, and causing changes in the morphology of the patient response waveform. This different response suggests a higher degree of statistic independence in sick patients when compared to healthy individuals regarding the patient response to the visual saccadic stimulus. We processed electro-oculogram records of six patient diagnosed with severe ataxia SCA2 and six healthy subjects used as control, employing independent component analysis (ICA), significant differences have been found in the statistical independence of the person response with the stimulus for 60° saccadic tests.

Keywords: Biomedical engineering, computer aided diagnosis, independent component analysis, ataxia, SCA2, electro-oculography.

1 Introduction

The autosomal dominant cerebellar ataxias (ADCAs) are a heterogeneous group of dominantly inherited neurological disorders characterized by progressive ataxia that results from degeneration of the cerebellum and its afferent and efferent connections. In most families there is clinical and neuropathological evidence of additional involvement of brainstem, basal ganglia, spinal cord, and the peripheral nervous system [1].

In Cuba, there are almost 800 patients and 8000 presymptomatic relatives at risk of developing some ADCA in the next few years. The most frequent molecular form is SCA2 (spino cerebellar ataxia type 2), with a prevalence of 43 cases per 100,000 inhabitants in Holguín province, which is the highest one reported worldwide, reaching 142 cases per 100,000 inhabitants in Baguanos municipality [2][3][4]. This form occurs commonly in persons of Spanish ancestry in northeastern Cuba, a figure much higher than that found in western Cuba

or in other parts of the world. The high prevalence is probably the result of a founder effect, but might be due to an interaction between a mutant gene and an unidentified environmental neurotoxin [24].

Several studies have reported oculomotor abnormalities in ADCA [14,5,6,7,8], slowness of saccades has been suggested as a relatively characteristic finding in SCA2[4,8], in particular in the Cuban ataxia SCA2 have been found remarkable slowness of the peak saccadic velocity at 60°, saccadic hypermetric deviation for stimulation angles of 10°, 20° and 30°, otherwise hypometric deviation is present at 60°, and injuries in the saccade onset, with abnormal latency increment, at all stimulation angles [4,9]. The above findings suggest significant differences in saccade morphology between healthy individuals and patients with SCA2, mainly for 60° of stimulus amplitude. The electro-oculographical records are quite different in healthy individuals and patients with a severe ataxia (Fig. 1).

The ocular movement records have been widely used in processing and classification of biological signals and pathological conditions: clinical sleep scoring [10,11], cerebellar dysfunctions [12,13,14], diagnosis of the visual system [15,16], amongst others, also in human computer interface and visual guided devices [17,18,19].

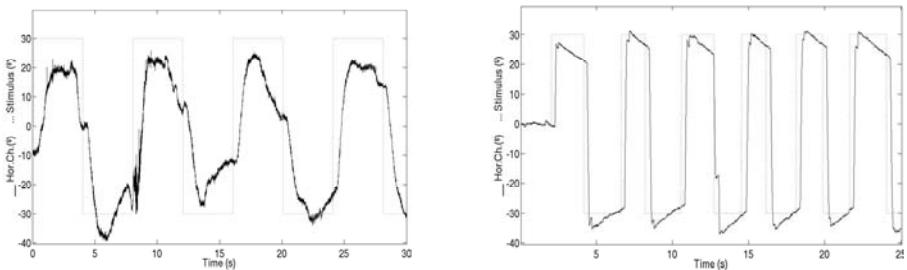


Fig. 1. Original records of horizontal saccade movements at 60°, in a patient of severe ataxia (left) and a healthy subject (right)

2 ICA Based Diagnosis of SCA-2

2.1 Why Use ICA?

Concisely, independent component analysis is aimed to find a linear transformation given by a matrix \mathbf{W} , so that the random variables $y_i, (i = 1, \dots, n)$ of $\mathbf{y} = [y_1, \dots, y_n]$ are as independent as possible in:

$$\mathbf{y}(t) = \mathbf{W} \cdot \mathbf{x}(t) \quad (1)$$

This linear blind source separation approach is suitable for the signals obtained by the EOG, as well as in other medical analysis such as electroencephalography (EEG), electrocardiography (ECG), magneto-encephalography (MEG), and functional magnetic resonance imaging (fMRI) [20,21,22,23,24,25,26].

Roughly speaking, in the analysis of EOG oriented to the detection of SCA2 (See Section 1), experts anticipate two possible behaviors of the individuals: sick and healthy conduct. During an experiment over a healthy subject, the horizontal movement of the eye is expected to follow the stimulus signal. Therefore, the horizontal eye movement and the stimulus will hold a direct dependence between them, i.e. the signals are not independent. In contrast, a sick individual may present a more chaotic response, depending on the severity of the disease. Consequently, the subject response will not depend in such a high degree on the stimulus signal, and the signals are independent (or at least, “not so dependent”).

To sum up, our approach uses independent component analysis as a classification algorithm criterion: if the independence measure (normally mutual information) reveals independence between the individual response and the stimulus signal, then it is rather possible that the individual presents some degree of ataxia or related disease.

2.2 Description of the Diagnosis Algorithm

The proposed algorithm for SCA-2 diagnosis will go along the following steps:

1. Set both horizontal response and stimulus signal in the same phase, i.e. correct the delay between the stimulus change and the saccade.
2. Normalize signals (x).
3. Apply ICA algorithm. Any well known ICA algorithm may be applied at this point (FastICA [27], Jade [28], GaBSS [29,30], etc.).
4. Normalize estimations (y)
5. Calculate error measure between estimations (y) and mixtures (x), according to the following expression:

$$RMSE(\mathbf{x}_i, \mathbf{y}_i) = \sqrt{\frac{\sum_{t=0}^N [x(t) - y(t)]^2}{N}} \quad (2)$$

6. Depending on the obtained error measure, a simple categorization algorithm (such as C-means) may be applied in order to classify individuals. Otherwise, an human expert may help in subject categorization based on the ICA results.

3 Experimental Results

3.1 Experiment Setup

The electro-oculogram recordings of six patients with severe ataxia and six healthy subjects diagnosed and classified in the “Centro de Investigación y Rehabilitación de las Ataxias Hereditarias de Holguín” (Centre for the Research

and Rehabilitation of Hereditary Ataxias, CIRAH) were used in order to perform the analysis of repeated ocular saccadic movement tests for 10°, 20°, 30° and 60° divergence stimuli.

All the records were carried out by the medical staff of CIRAH. Each individual was placed in a chair, with a head fixation device to avoid head movements, the variables were collected by a two channel electronystagmograph (Otoscreen, Jaeger-Toennies, D-97204 Hchberg, Germany). Recording conditions were set as follows: electrodes of silver chloride placed in the external borders of right eye

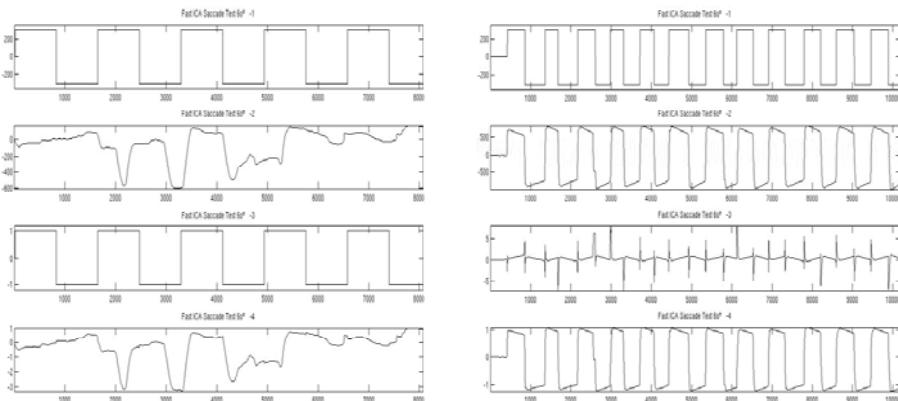


Fig. 2. Stimulus (1), response (2) and ICA components (3 and 4) obtained at 60° of stimulation for patients (left) and control subjects (right)

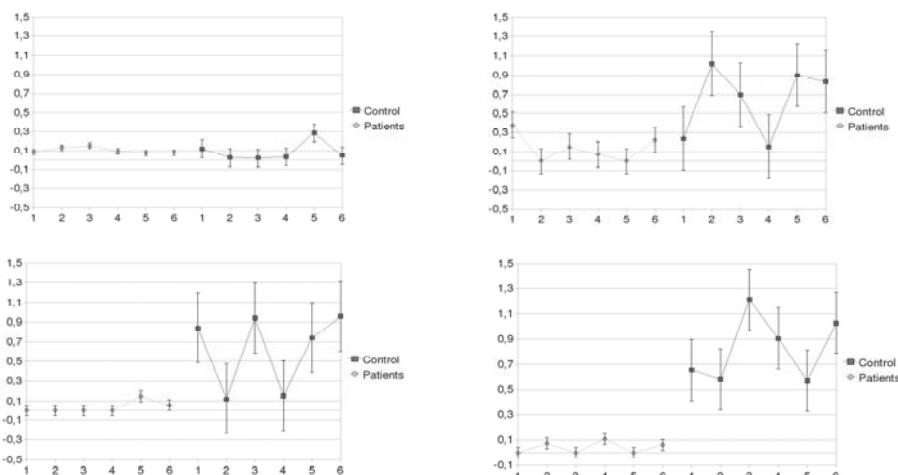


Fig. 3. Error average and standard deviation for patients and control subjects with stimulation angles of 10° (1), 20° (2), 30° (3) and 60°(4)

(active electrode) and left eye (reference electrode), high pass filtering 0.002 Hz, low pass filtering 20 Hz, sensitivity 200 μ V / division, and sampling frequency 200 Hz. For stimulus generation a black screen CRT display showing a white circular target with an angular size of 0.7° was used. The stimulus and patient response data are automatically stored in ASCII files by Otoscreen electroneurotagmograph.

3.2 Results

The patient response was filtered using a median filter, to obtain a clean waveform of the patient response, then it was phased with the stimulus. Finally FastICA was applied to get the independent components.

A visual inspection at the relationship between mixtures (stimulus and response) and ICA estimations obtained for every subject for stimulation angles of 10°, 20° and 30° shows a chaotic behavior for patients and control subjects. If the stimulation is done at 60°, our approach applied over sick patients produces components very similar to the original mixtures. This is due to the fact that the patient response and the visual stimulus are already statistically independent. In contrast, for control subjects a mixture of stimulus and response is obtained

Table 1. Consolidated mean and standard deviation of error for every angle (after 15 executions of each record)

RMS Error	10°		20°		30°		60°	
	Control	Patients	Control	Patients	Control	Patients	Control	Patients
Mean	0,18	0,23	0,86	0,38	0,82	0,32	1,04	0,4
Standard Dev.	0,43	0,25	0,38	0,35	0,42	0,33	0,26	0,26

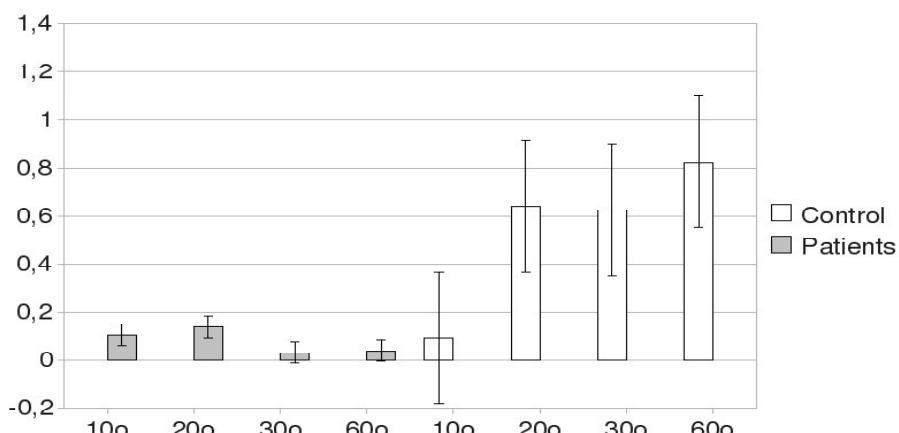


Fig. 4. Consolidated mean and standard deviation of error for every angle in control and patients subjects

for each component. As it is shown in Fig. 2, more uniform results are obtained for the most similar component to the stimulus.

The errors calculated for this component for every individual illustrate an overlapping between patients and control subjects, as it is shown in Fig. 3, except for the stimulation angle of 60°, where a remarkable difference is obtained, in correspondence with the visual inspection described above.

This behavior is consolidated in Table II, the errors were averaged for patients and control subjects, the mean for 60° shows the highest value, and considering the standard deviation there is no overlapping between both groups for this stimulation angle (See Fig. 4).

4 Conclusions

This article discusses a satisfactory approach for SCA2 classification using independent component analysis. The proposed method starts from the assumption that the response to a visual stimulus is different in a healthy individual when compared to the response of an individual afflicted by SCA-2. In the later situation, the response from the individual is not dependent on the visual stimulus, so that the ICA algorithm estimations will be similar to the obtained observations. This criterion has shown to be suitable in order to distinguish between sick (patients) and healthy (control) individuals.

Future research will focus on automatic classification and diagnosis from the observed EOG signals, starting from the results achieved by this approach. Also, a more specific diagnosis based on the stage of the disease will centre forthcoming investigation.

Acknowledgments. The authors would like to thank the rest of the personnel in the Centre for the Research and Rehabilitation of Hereditary Ataxias “Carlos J. Finlay”, Holguín, (Cuba) for their support and collaboration. This work has been partially supported by the Spanish MAEC-AECID fellowship program (2008).

References

- Burk, K., Fetter, M., Abele, M., Laccone, F., Brice, A., Dichgans, J., Klockgether, T.: Autosomal dominant cerebellar ataxia type i: oculomotor abnormalities in families with sca1, sca2, and sca3. *Journal of Neurology* 246(9), 789–797 (1999)
- Orozco, G., Estrada, R., Perry, T.L., Araa, J., Fernandez, R., Gonzalez-Quevedo, A., Galarraga, J., Hansen, S.: Dominantly inherited olivopontocerebellar atrophy from eastern cuba: Clinical, neuropathological, and biochemical findings. *Journal of the Neurological Sciences* 93(1), 37–50 (1989)
- Pérez, L.V., Cruz, G.S., Ochoa, N.C., Labrada, R.R., Díaz, J.R., Mederos, L.A., Mesa, J.L.: Electrophysiological features in patients and presymptomatic relatives with spinocerebellar ataxia type 2. *Journal of the Neurological Sciences* 263(1-2) (2007)

4. Velázquez, L.: Ataxia espino cerebelosa tipo 2. Principales aspectos neurofisiológicos en el diagnóstico, pronóstico y evaluación de la enfermedad. Ediciones Holguín, Holguín (2006)
5. Rivaud-Pechoux, S., Drr, A., Gaymard, B., Cancel, G., Ploner, C.J., Agid, Y., Brice, A., Pierrot-Deseilligny, C.: Eye movement abnormalities correlate with genotype in autosomal dominant cerebellar ataxia type i. *Annals of Neurology* 43(3), 297–302 (1998)
6. Hutton, J.T., Albrecht, J.W., Kuskowski, M., Schut, L.J.: Abnormal ocular motor function predicts clinical diagnosis of familial ataxia. *Neurology* 37(4), 698–701 (1987)
7. Yamamoto, H., Sobue, I.: Analysis of the abnormal ocular movements in spinocerebellar degenerations and the influence of the thyrotropin releasing hormone. *Clinical Neurology* 20(8), 637–645 (1980)
8. Klostermann, W., Zhinke, C., Heide, W., Kmpf, D., Wessel, K.: Slow saccades and other eye movement disorders in spinocerebellar atrophy type 1. *Journal of Neurology* 244(2), 105–111 (1997)
9. Velázquez-Pérez, L., Almaguer-Mederos, L., Santos-Falcón, N., Hechavarría-Pupo, R., Sánchez-Cruz, G., Panque-Herrera, M.: Spinocerebellar ataxia type 2 in cuba. a study of the electrophysiological phenotype and its correlation with clinical and molecular variables. *Revista de Neurología* 33(12), 1129–1136 (2001)
10. Virkkala, J., Hasan, J., Vrri, A., Himanen, S.L., Mller, K.: Automatic sleep stage classification using two-channel electro-oculography. *Journal of Neuroscience Methods* 166(1), 109–115 (2007)
11. Estrada, E., Nazeran, H., Barragan, J., Burk, J.R., Lucas, E.A., Behbehani, K.: Eog and emg: Two important switches in automatic sleep stage classification. In: 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. EMBS 2006, pp. 2458–2461 (2006)
12. Spicker, S., Schulz, J.B., Petersen, D., Fetter, M., Klockgether, T., Dichgans, J.: Fixation instability and oculomotor abnormalities in friedreich's ataxia. *Journal of Neurology* 242(8), 517–521 (1995)
13. Yokota, T., Hayashi, H., Hirose, K., Tanabe, H.: Unusual blink reflex with four components in a patient with periodic ataxia. *Journal of Neurology* 237(5), 313–315 (1990)
14. Hubner, J., Sprenger, A., Klein, C., Hagenah, J., Rambold, H., Zuhlke, C., Kompf, D., Rolfs, A., Kimmig, H., Helmchen, C.: Eye movement abnormalities in spinocerebellar ataxia type 17 (sca17). *Neurology* 69(11), 1160–1168 (2007)
15. Güven, A., Kara, S.: Classification of electro-oculogram signals using artificial neural network. *Expert Systems with Applications* 31(1), 199–205 (2006)
16. Irving, E.L., Steinbach, M.J., Lillakas, L., Babu, R.J., Hutchings, N.: Horizontal saccade dynamics across the human life span. *Invest. Ophthalmol. Vis. Sci.* 47(6), 2478–2484 (2006)
17. Kumar, D., Poole, E.: Classification of eog for human computer interface. In: Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings, vol. 1, pp. 64–67 (2002)
18. Brunner, S., Hanke, S., Wassertheuer, S., Hochgatterer, A.: Eog pattern recognition trial for a human computer interface. Universal Access in Human-Computer Interaction. Ambient Interaction, 769–776 (2007)
19. Komogortsev, O.V., Khan, J.I.: Eye movement prediction by kalman filter with integrated linear horizontal oculomotor plant mechanical model. In: Proceedings of the 2008 symposium on Eye tracking research and applications, pp. 229–236. ACM, New York (2008)

20. Vaya, C., Rieta, J.J., Sanchez, C., Moratal, D.: Convulsive blind source separation algorithms applied to the electrocardiogram of atrial fibrillation: Study of performance. *IEEE Transactions on Biomedical Engineering* 54(8), 1530–1533 (2007)
21. Anemuller, J., Sejnowski, T.J., Makeig, S.: Complex independent component analysis of frequency-domain electroencephalographic data. *Neural Networks* 16(9), 1311–1323 (2003)
22. Funase, A., Yagi, T., Barros, A.K., Cichocki, A., Takumi, I.: Single trial analysis on saccade-related eeg signal. In: 3rd International IEEE/EMBS Conference on Neural Engineering. CNE 2007, pp. 371–374 (2007)
23. Hironaga, N., Haruhana, K., Liu, L.C., Fenwick, P.B.C., Ioannides, A.A.: Monitoring of eye movement and its use for artifact elimination. *International Congress Series* 1270, 134–137 (2004)
24. Milanesi, M., Martini, N., Vanello, N., Positano, V., Santarelli, M., Landini, L.: Independent component analysis applied to the removal of motion artifacts from electrocardiographic signals. *Medical and Biological Engineering and Computing* 46(3), 251–261 (2008)
25. Guilhon, D., Medeiros, E., Barros, A.K.: Ecg data compression by independent component analysis. In: IEEE Workshop on Machine Learning for Signal Processing, pp. 189–193 (2005)
26. Kocyigit, Y., Alkan, A., Erol, H.: Classification of eeg recordings by using fast independent component analysis and artificial neural network. *Journal of Medical Systems* 32(1), 17–20 (2008)
27. Hyvarinen, A., Oja, E.: A fast fixed-point algorithm for independent component analysis. *Neural Computation* 9(7), 1483–1492 (1997)
28. Cardoso, J.F.: Source separation using higher order moments. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, vol. 4, pp. 2109–2112 (1989)
29. Rojas, F., Alvarez, M.R., Puntonet, C.G., Martin-Clemente, R.: Applying neural networks and genetic algorithms to the separation of sources. In: Garijo, F.J., Riquelme, J.-C., Toro, M. (eds.) IBERAMIA 2002. LNCS (LNAI), vol. 2527, pp. 420–429. Springer, Heidelberg (2002)
30. Martin-Clemente, R., Puntonet, C.G., Rojas, F.: Post-nonlinear blind source separation using methaheuristics. *Electronics Letters* 39(24), 1765–1766 (2003)

Hippocampus, Amygdala and Basal Ganglia Based Navigation Control

Ansgar Koene and Tony J. Prescott

Adaptive Behaviour Research Group, Sheffield University,
Western Bank, S10 2TP Sheffield, UK
{A.Koene,T.J.Prescott}@sheffield.ac.uk
<http://www.abrg.group.shef.ac.uk/>

Abstract. In this paper we present a novel robot navigation system aimed at testing hypotheses about the roles of key brain areas in foraging behavior of rats. The key components of the control network are: 1. a Hippocampus inspired module for spatial localization based on associations between sensory inputs and places; 2. an Amygdala inspired module for the association of values with places and sensory stimuli; 3. a Basal Ganglia inspired module for the selection of actions based on the evaluated sensory inputs. By implementing this Hippocampus-Amygdala-Basal Ganglia based control network with a simulated rat embodiment we intend to test not only our understanding of the individual brain areas but especially the interaction between them. Understanding the neural circuits that allows rats to efficiently forage for food will also help to improve the ability of robots to autonomously evaluate and select navigation targets.

Keywords: Action selection, navigation, biologically inspired, Hippocampus, Amygdala, Basal Ganglia, place value association.

1 Introduction

Efficient foraging behavior relies on a combination of spatial cognition, motivation, and goal-directed navigation in order to maximize the chances of quickly finding sufficient food. Choosing an optimal foraging route requires an evaluation of the relative worth of potential food locations as well as the ability to successfully navigate to the chosen locations.

In this study we combined neurophysiological data and computational neuroscience methodologies to develop a better understanding of the brain systems underlying spatial representation and decision-making in foraging rats and construct a novel robot navigation system based on this understanding. Specifically we focused on the hippocampus-amamygdala-basal ganglia complex of rats.

Electrophysiological experiments with rats led to the discovery that in a subset of neurons in the hippocampal region the firing rate was correlated with the location of the animal in a test environment [23] and that hippocampal damage causes spatial learning deficits [4]. The representation encoded by these *place*

cells integrates the relationships among visual cues with kinesthetic self-motion information in order to recognize previously visited places and distinguish among perceptually similar places [5].

The amygdala system evaluates the innate or conditioned value of environmental cues. Context dependent conditioning is impaired by both hippocampus and amygdala lesions, but simple stimulus conditioning is impaired only by lesions of the amygdala [6]. Evidence suggesting a central role for the amygdala in discriminating the magnitude of reward comes from studies [7] in which rats with lesions of the central nucleus of the amygdala failed to discriminate between arms of a radial-maze that contained one or seven pieces of food. Lesions of the lateral amygdala made after the formation of cue-reward associations eliminated the conditioned preference [8] indicating that the lateral amygdala is also critical for the expression of value associations after they have been acquired.

The basal ganglia are a group of highly interconnected central brain structures that acts as action selection mechanism resolving conflicts between functional units that are in competition for behavioral expression [11,12]. Basal ganglia input occurs via a series of topographically organized, parallel processing streams [1] that encode the salience of potential actions. The action selection is done by maintaining or increasing inhibition on undesired actions and releasing inhibition from desired actions [9,10].

Computational models of each of these brain areas were integrated with sensory processing and motor execution modules to produce a navigation control system for a simulated robot rat. In order to facilitate flexible interaction between each module the BRAHMS framework [22] was used to handle the communication between the brain system models as well as the sensors and effectors of the simulated animal.

The performance of our hippocampus-amygadala-basal ganglia based navigation control system was tested by replicating a reinforcement based plus-maze learning task that has been used in a series of neurophysiological studies with rats [19,20,21]. Importantly, this plus maze task involves: 1. distinguishing between rewarded and non-rewarded locations based on visual cues; 2. place recognition based on the configuration of visual cues outside the plus-maze; 3. learning of specific place-value associations based on place specific reinforcement magnitudes; 4. selection of approach behaviors toward simultaneously available rewards whose relative, place-specific, magnitudes were learnt on previous trials.

2 System Architecture

The overall architecture of the navigation control system is shown in figure 1. There are two parallel processing streams. The stream going through the *dorsal* Basal Ganglia module processes inputs from touch (whiskers) and taste (battery recharge) sensors that elicit fixed stereotypical responses, i.e. collision avoidance reflexes and food ingestion. The stream going through the *ventral* Basal Ganglia module processes inputs from the visual and kinesthetic self-motion senses that guide goal-directed navigation.

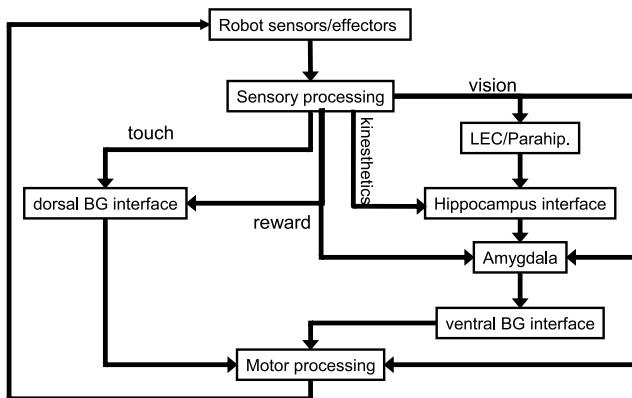


Fig. 1. Architecture of the navigation control system. The path through the *dorsal Basal Ganglia* controls fixed action patterns such as collision avoidance. The path through the *ventral Basal Ganglia* controls stimulus value dependent orienting responses.

2.1 Modules

Robot Sensors/Effectors. The simulated robot in our experiment is a wheeled rat-like robot (see figure 2 A). The robot effectors are its two independently driven wheels that allow it to move in the simulated environment. The sensory capabilities of the robot are:

vision provided by 2x160° color cameras place on the left and right side of the rat head giving a 320° visual field with a 40° blind area in the back (similar to a real rat [13]).

touch left and right whisker-like sensors for detecting left, right and frontal contact with the maze walls.

kinesthetic self motion sense provided by wheel motion sensors producing proprioception-like feedback of self motion.

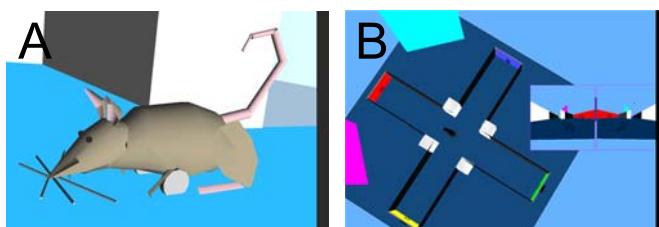


Fig. 2. Simulated wheeled rat-like robot (A) and plus-maze environment (B) used in our experiment

kinesthetic orientation sense implemented using a gyroscope based rotation sensor for measuring orientation changes, similar to the vestibular system in animals.

reward detection a voltmeter indicating the current charge in the robot battery. Since the "food" reward is simulated by a battery recharge this allows the robot to detect when and how much reward it receives.

Sensory Processing. The only biological constraint imposed on this module is that the outputs contain no information that would be impossible for a rat to have. One of the main tasks of the sensory processing module is to translate the modality specific sensory input signals into salience signals with a common normalized range of values across all modalities. The sensory processing produces the following outputs:

vision: In order to avoid complex issues of scene segmentation and object recognition all spatial landmarks and target locations were designated with unique colors (see figure 2 B). The visual processing simply detects the mean location and number of pixels of each color in the visual field. In the real rat experiments the stimuli at the goal locations all looked identical such that they were distinguishable only by the relative locations of external landmarks. The output from the visual processing therefore discarded all color information producing an unlabeled vector of egocentrically perceived object directions and a corresponding vector of visual object sizes. Only the perceived directions and sizes of the external landmarks and the maze center were labeled since these had clearly distinguishable shapes in the rat experiments.

touch: Input signals from the tactile sensors are converted into a three element vector signaling the salience of touch (degree of whisker deformation) at the left, right and front.

kinesthetic senses: The kinesthetic inputs concerning position and orientation changes are used to produce an estimate of current position relative to a starting reference by means of path integration. At the start of each simulated experiment the robot was placed at the center of the plus-maze with a random orientation. From there it visually located the two external landmarks and designated the direction between the landmarks as the 0° direction. Path integration is subsequently achieved by simply summing the orientation and position changes over time.

reward detection: The magnitude of reward that is received at any one time point (simulation iteration) is derived from the change in battery charge with respect to the previous time point.

Parahippocampus. The parahippocampus transforms egocentrically perceived object directions into self-orientation independent signals that can be used to recognize spatial locations based on the perceived spatial configuration of visible objects [14]. The output from parahippocampus is a vector specifying the visual angles between the external landmarks and each visual object in the maze.

Hippocampus. For spatial navigation tasks the primary function of the hippocampus is to map the environment and estimate current self-location. Spatial mapping relies on the inputs from the parahippocampus module concerning perceived spatial stimulus configurations and the current place estimate from path integration. The parahippocampal inputs are first compared against the previously mapped stimulus configurations. If no good match is found, a new *place cell* is created associating the inputs from parahippocampus with the path integration based place estimate. If however good match(es) are found, the hippocampal and path integration based place estimate are compared. If these place estimates are very different it is assumed that path integration has drifted due to accumulation of small errors and the place estimate is adjusted accordingly.

For goal directed navigation we hypothesize a second functional role for the hippocampus. Using the same place-stimulus associations that are generated during spatial mapping the hippocampus may be involved in distal place recognition. A possible neural substrate for this might be the spatial view cells that have been reported in primate hippocampus [15]. Even though the exact visual angles between the external landmarks and the objects inside the maze is specific to each maze location the gross distribution of the landmark positions with respect to the stimuli remains mostly invariant. For one stimulus light both landmarks are on the left, for another they are both on the right etc. We propose that it is these secondary place-stimulus association matches for distal locations that produce the inputs from the hippocampus to the amygdala.

Amygdala. The amygdala module provides association of values, i.e. salience of attraction or repulsion, with basics sensory stimuli or hippocampal place inputs. Stimulus/place-value associations are established whenever an innately rewarding/punishment related input is received. Any sensory input that is present at the time of the reward becomes associated with the rewarding input. For instance, when food is found near light, light becomes associated with food. At the same time an association is made between the current self-position estimate, from the hippocampus, and the food stimulus. The magnitude of the stimulus/place-value is determined by the strength of the association with the reward signal, which in turn is determined by the size of the reward that was received. The value associated with sensory stimuli/places is then assigned to the egocentric direction in which the stimuli/places are perceived.

Dorsal/Ventral Basal Ganglia. The dorsal and ventral basal ganglia modules are essentially the same. The only difference is the information that is processed in the modules and the type of behaviors they control.

In the dorsal module the basal ganglia channels represent *fixed action patterns* (FAPs). FAPs are species-specific, instinctive responses to specific patterns of stimulation [23]. A distinctive feature is that, once elicited, the overall form of the pattern is uninfluenced by further external cues [24]. The four FAPs in our navigating robot are: moving away from a wall on the right (1) on the left (2) or in front (3) and staying at the current location to consume food (4).

In the ventral module the basal ganglia channels represent *orienting responses*. In order to limit the number of necessary channels in the ventral module, the spatial directions are limited to eight equally spaced directions (forward, forward-right, right, backward-right, backward, backward-left, left and forward-left). All saliences assigned by the amygdala to directions within one of these eight sectors are pooled using the *max* operator.

The basal ganglia model used in these modules is an implementation of the model published in [16] which was previously applied in a robotic controller in [17,18]. The outputs of the basal ganglia are inhibitory in nature and function to suppress undesired actions [9]. Action selection therefore takes the form of selective dis-inhibition of the action that is coded by the channel with the most salient input, akin to an inverse winner-takes-all where the winner is the only channel that is not active.

In order to insure that the dorsal and ventral basal ganglia do not simultaneously disinhibit competing actions a high gain copy of the most salient signal to the dorsal pathway is used as an additional input to the ventral path. Thus, if there is a significant input for triggering a collision avoidance or reward collection behavior the navigation behavior is overridden.

Motor Processing. The motor processing module translates actions (as selected by the basal ganglia modules) into motor commands (left & right wheel velocities).

Since the dorsal path controls fixed action patterns a look-up table is used to execute these actions. If, for instance, the action to *move away from a wall on the left* is selected, the corresponding FAP is disinhibited in the look-up table causing a weak backward motion in the left wheel and a strong backward motion in the right wheel.

For orienting behaviors, selected by the ventral basal ganglia, each direction where an object is perceived (as signaled by the sensory processing module) produces a *potential movement command*. The ventral basal ganglia output inhibits all *potential movement commands* that are not in the selected direction sector. The visual salience weighted sum of the non-inhibited *potential movement commands* is used to determine the speeds of the left and right wheel motion.

Finally, if the basal ganglia modules produce no clear action selection (none or more than one action is disinhibited) right and left wheel velocities are set to zero and a *restlessness* level starts to build up. When restlessness reaches a pre-determined threshold, the robot performs a random left or right turn with some forward motion.

3 Simulator and Environment

The rat robot and plus maze environment were simulated using the Webots¹ robot simulation toolkit. The robot and environment are shown in figure 2. The

¹ Cyberbotics Ltd, www.cyberbotics.com

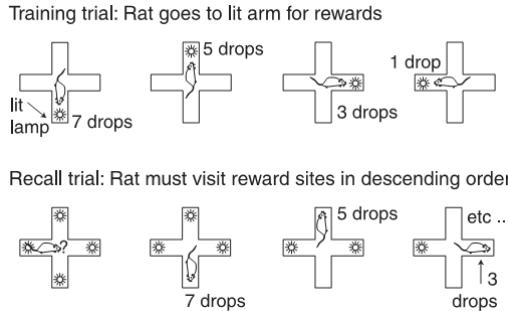


Fig. 3. The experimental task. First the robot performed a series of training trials where the correct choice was guided by the lit stimulus object light at the end of the appropriate maze arm. Each trial comprised a sequence of visits to the ends of the four maze arms providing battery "food" rewards with magnitude 7, 5, 3 and 1. During recall trials all object lights were lit, then were turned off one by one as the robot visited the reward locations at the end of the maze arms in the same order of descending reward value. (Adapted from Tabuchi et al., 2000).

colored panels at the end of the maze arm represent the stimulus object lights. When the robot touches the maze end walls the colored panel in that maze arm turns gray (i.e. the light is deactivated). A touch sensor in the floor of the maze center detects when the robot has returned to the center area and triggers the next phase of the task.

4 An Evaluation Task from Experimental Neurobiology

To test our rat brain inspired robot navigation system we used the *differentially rewarded plus-maze task* that was previously used by [19,20,21] in real rat experiments on the roles of amygdala, hippocampus and basal ganglia in spatial navigation. The basic task is illustrated in figure 3. The association between stimulus lights and reward was assumed to be pre-conditioned thus the Amygdala module was pre-coded with a strong value for lit objects. After reaching the end of a maze arm, and thus extinguishing the corresponding object light, the robot was motivated to return to the center of the maze since this was then the only lit object visible inside the maze. Upon reaching the maze center the light on the next arm was lit/no-longer occluded by the maze walls.

5 Results: Task Performance

A video of the simulated robot successfully performing the plus-maze task is available at <http://www.abrg.group.shef.ac.uk/people/ansgar/>. The simulation consisted of three stages corresponding to the pre-training, training and post-training sessions in the rat experiments:

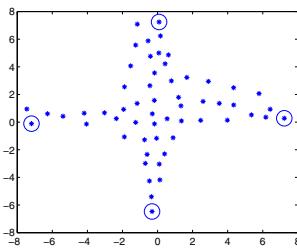


Fig. 4. Hippocampal place map of plus-maze learnt by the robot. Places encoded by the place cells are indicated with asterisks symbols. Places associated with values in the Amygdala are indicated with circles.

1. a pre-training test phase (3000 time steps) in which the stimuli are identical to the *recall trial* (figure 3, lower panel) but where the robot has had no prior exposure to the plus-maze and the only value association in amygdala is the pre-conditioned value for light cues. During this time the order in which the maze is explored is based simply on the orientation the robot happens to be facing at the start of the simulation. During this maze exposure the robot maps the plus-maze with its hippocampal place-cells.
2. a training phase (3500 time steps) during which the robot is guided down each maze arm in turn and given location specific amounts of reward when the end of an arm is reached. During this time the robot acquires the location specific object-value associations in the amygdala.
3. a post-training test phase (3750 time steps), or recall trial, during which the robot uses the learnt value associations to visit each plus maze arm in descending order of reward value.

The total duration of the simulation was 10250 time steps. The basal ganglia achieved clean action selection on 96.2% of occasions (390 time steps). During the pre-training test phase clear selection occurred 94.5% of the time (164 time steps) while after training this was increased to 98.3% of the test phase duration (62 time steps).

During 37% of the pre-training period the action selections were closely fought (1111 time steps), i.e. the level of inhibition on the second most salience action was less than half the resting level inhibition. In the post-training period closely fought decisions occurred only 22% of the time (818 time steps).

There were 1841 time steps during the total experiment where at least one of the external landmarks was occluded making visual place recognition impossible since visual place information is encoded as perceived angles between the external landmarks and the visual objects in the maze. During these time steps the amygdala is unable to associate the sensory stimuli with place specific values. In 78 instances this was correlated with indecision. In 498 instances the visual information was not relevant at that time since the robot was responding to a collision or feeding impulse with action selection being processed by the *dorsal* basal ganglia. For the other time steps during which a landmarks was occluded

action selection did not require the amygdala module since only one target object was visible to the robot at that time.

Figure 4 shows the hippocampal place map, illustrating the distribution of places coded by the place cells (asterisks symbols). Circles indicate the place cells that are associated with values in the amygdala.

6 Discussion

We have described a robot navigation control system based on the Hippocampus-Amygdala-Basal Ganglia circuit that plays a critical role in navigation and foraging behavior of rats. This controller selects between navigation goals based on spatial context defined values and guides the robot towards the selected goal.

As expected from previous implementations of Basal Ganglia models for robot control [18] our Basal Ganglia modules switch effectively between competing actions/targets depending on their relative salience.

The Hippocampus modules successfully integrated sensory information from visual and self-motion senses to establish a sense of self-location with respect to the surroundings and provide this place information as contextual cue for disambiguating visually identical stimuli.

The Amygdala module in turn used this contextual information to modulate the salience of sensory inputs thereby guiding the action selection in the Basal Ganglia toward the most highly rewarded stimulus. In addition to solving the plus-maze task this had the added benefit of reducing the number of instances of indecision by more than half.

In future we plan to use this model to test our understanding of these navigation related brain areas. In order to do this we will update the implementations of the Hippocampus and Amygdala modules to increase their neurophysiological accuracy in order to enable simulated electrophysiology and lesion studies.

Acknowledgments. This work was supported by the European Union Framework 6 IST project 027819 (ICEA project: www.iceaproject.eu).

References

1. Alexander, G.E., Crutcher, M.D.: Functional architecture of basal ganglia circuits: neural substrates of parallel processing. *Trends in Neurosciences* 13(7), 266–271 (1990)
2. O’Keefe, J., Dostrovsky, J.: The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely moving rat. *Brain Research* 34, 171–175 (1971)
3. O’Keefe, J., Conway, D.H.: Hippocampal place units in the freely moving rat: Why they fire when they fire. *Experimental Brain Research* 31, 573–590 (1978)
4. Morris, R.G., Garrud, P., Rawlins, J.N., O’Keefe, J.: Place navigation impaired in rats with hippocampal lesions. *Nature* 24, 681–683 (1982)
5. Jeffery, K.J., O’Keefe, J.M.: Learned interaction of visual and idiothetic cues in the control of place field orientation. *Experimental Brain Research* 127, 151–161 (1999)

6. Phillips, R.G., LeDoux, J.E.: Differential contribution of amygdala and hippocampus to cued and contextual fear conditioning. *Behavioral Neuroscience* 106, 274–285 (1992)
7. Kesner, R.P., Walser, R.D., Winzenried, G.: Central but not basolateral amygdala mediates memory for positive affective experiences. *Behavioural Brain Research* 33, 189–195 (1989)
8. Everitt, B.J., Morris, K.A., O'Brien, A., Robbins, T.W.: The basolateral amygdala-ventral striatal system and conditioned place preference: Further evidence of limbic-striatal interactions underlying reward-related processes. *Neuroscience* 42, 1–18 (1991)
9. Mink, J.W.: The basal ganglia: Focused selection and inhibition of competing motor programs. *Progress in Neurobiology* 50(4), 381–425 (1996)
10. Wickens, J.: Basal ganglia: structure and computations. *Network-Computation in Neural Systems* 8(4), R77–R109 (1997)
11. Prescott, T.J., Redgrave, P., Gurney, K.N.: Layered control architectures in robots and vertebrates. *Adaptive Behavior* 7(1), 99–127 (1999)
12. Redgrave, P., Prescott, T., Gurney, K.N.: The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience* 89, 1009–1023 (1999)
13. Burn, C.C.: What is it like to be a rat? Rat sensory perception and its implications for experimental design and rat welfare. *Applied Animal Behaviour Science* 112, 1–32 (2008)
14. Bird, C.M., Burgess, N.: The hippocampus and memory: insights from spatial processing. *Nature Reviews Neuroscience* 9, 182–194 (2008)
15. Rolls, E.R., Stringer, S.M.: Spatial view cells in the hippocampus, and their idiothetic update based on place and head direction. *Neural Networks* 18(9), 1229–1241 (2005)
16. Humphries, M.D., Gurney, K.N.: The role of intra-thalamic and thalamocortical circuits in action selection. *Network: Computation in Neural Systems* 13, 121–156 (2002)
17. Girard, B., Cuzin, V., Guillot, A., Gurney, K.N., Prescott, T.J.: A basal ganglia inspired model of action selection evaluated in a robotic survival task. *Journal of Integrative Neuroscience* 2(2), 179–200 (2003)
18. Prescott, T.J., Montes Gonzalez, F.M., Gurney, K., Humphries, M.D., Redgrave, P.: A robot model of the basal ganglia: Behavior and intrinsic processing. *Neural Networks* 19, 31–61 (2006)
19. Albertin, S.V., Mulder, A.B., Tabuchi, E., Zugaro, M.B., Wiener, S.I.: Lesions of the medial shell of the nucleus accumbens impair rats in finding larger rewards, but spare reward-seeking behavior. *Behav. Brain Res.* 117, 173–183 (2000)
20. Tabuchi, E., Mulder, A.B., Wiener, S.I.: Position and behavioral modulation of synchronization of hippocampal and accumbens neuronal discharges in freely moving rats. *Hippocampus* 10, 717–728 (2000)
21. Khamassi, M., Mulder, A.B., Tabuchi, E., Douchamps, V., Wiener, S.: Anticipatory reward signals in ventral striatal neurons of behaving rats. *European Journal of Neuroscience* 28, 1849–1866 (2008)
22. BRAHMS Modular Execution (Simulation) Framework,
<http://brahms.sourceforge.net/>
23. Lorenz, K.: Der kumpan in der umwelt des vogels. *Journal of Ornithology* 83, 137–213 (1935)
24. Colgan, P.: *Animal Motivation*. Chapman and Hall, London (1989)

A Framework for Simulation and Analysis of Dynamically Organized Distributed Neural Networks

Vladyslav Shaposhnyk^{1,2}, Pierre Dutoit^{1,3}, Victor Contreras-Lámus¹,
Stephen Perrig³, and Alessandro E.P. Villa^{1,3,4}

¹ Neuroheuristic Research Group, Grenoble Institute of Neuroscience,
Université Joseph Fourier, Grenoble, France

² Non-linear Analysis Department, Institute for Applied System Analysis,
State Technical University “Kyivskyy Politehnichnyy Instytut”, Kiev, Ukraine

³ Sleep Research Laboratory, Dept. of Psychiatry Belle-Idée,
Hôpitaux Universitaires de Genève, Switzerland

⁴ Neuroheuristic Research Group, Information Science Institute,
University of Lausanne, Switzerland
<http://gin7.neuroheuristic.org>

Abstract. We present a framework for modelling and analyzing emerging neural activity from multiple interconnected modules, where each module is formed by a neural network. The neural network simulator operates a 2D lattice tissue of leaky integrate-and-fire neurons with genetic, ontogenetic and epigenetic features. The Java Agent DEvelopment (JADE) environment allows the implementation of an efficient automata-like virtually unbound and platform-independent system of agents exchanging hierarchically organized messages. This framework allowed us to develop linker agents capable to handle dynamic configurations characterized by the entrance and exit of additional modules at any time following simple rewiring rules. The development of a virtual electrode allows the recording of a “neural” generated signal, called electrochirogram (EChG), characterized by dynamics close to biological local field potentials and electroencephalograms (EEG). These signals can be used to compute Evoked Potentials by complex sensory inputs and comparisons with neurophysiological signals of similar kind.

Keywords: Spiking neural networks, hierarchical neural networks, distributed computing, computational neuroscience, bio-informatics.

1 Introduction

The brain represents by far the most complex organ of the human body and its simulation will certainly remain out of reach for a long time. However the principle that govern its development and processing represent a source of inspiration for the design of artifacts [1]. In principle the design would consist to create programs that reproduce cognitive processes directly at higher representational level

or to create *in silico* artificial neural network systems. The project PERPLEXUS is aimed at developing an ubiquitous, scalable and distributed platform dedicated to the simulation of large-scale self-organising networks and to the observation of potentially emerging behaviours [23]. This platform is composed of custom reconfigurable devices endowed with computing, behaving and communicating modules called *Ubidules*. They are based on a custom designed processor called *Ubichip* and are characterized by custom designed bio-inspired features such as growth, learning, and evolution.

This paper advocates that a network of Ubidules may offer an interesting platform to implement a network of dynamically interacting modules characterized by integrate-and-fire neuromimes. In particular we present the JUbiNet simulator of distributed neural networks developed in the frame of the PERPLEXUS project and some examples of its output in the form of brain-like recorded signals.

2 General Concepts

JUbiNet is a highly expandable and flexible framework aimed at simulating hierarchical neural systems. The framework is based upon three major levels, which are: phylogenetic, ontogenetic, and epigenetic. All components are organized in a modular way such to enable inter-operability, compatibility, and expandability of the system and its parts on all levels [4].

At the Phylogenetic level several simulated neural system features (neural network parameters, topology rules of distributed network, etc.) are encoded and stored in a *genome*, distributed to lower levels of the application. Selection of alternate values of the parameters (i.e., the *alleles*) is performed at this level associated to a computational neurogenetic modeling [5].

The Ontogenetic level describes the origin and the development of the system during its early stages of development. Genome decoding, neural network initialization, and inter-network connection establishment rules are performed within this conceptual level. The Epigenetic level refers to learning features, which are limited to an individual lifetime. The neural network simulator itself naturally fits in this layer.

The current version of JUbiNet carries a full implementation of the epi- and ontogenetic levels and partially the phylogenetic level. JUbiNet is provided with flexible configuration facilities, a collection of data processing objects and network handling that allows the simulation of customized spiking neural networks organized in topologies of interest.

The Bio-Inspired Neural Network Simulator. The simulator is designed to efficiently emulate neural network models with emphasis on facilities for model reconfiguration and adjustment and on functionally rich possibilities for detailed network state acquisition. The neural simulation consists in a set of processes run over a set of neurons.

The processes in the neural network fully determine the functional model that includes processes such as synaptogenesis, activity transmission, state recorder,

learning, etc. The simulator defines a set of interfaces to general neural concepts and property access routines, like: neuron, synapse, network, signal-processing routines, input/output routines. With predefined implementations of standard objects it is possible to assemble common neural network models. The interfaces provided by the simulator are designed to extend or replace all default objects by user defined ones.

The Distributed Hierarchical Framework. The simulation begins when a network is composed with agents (of software or hardware nature) running distributed networking modules. Those agents are waiting for genomic information, which is prepared and transmitted by simulation of planning or phylogenetic modules. Genome decoding triggers network initialization and the simulation starts when all systems are initialized.

The distributed network simulator is divided into four main parts: the network discovery system, the link manager system, the input/output mapping and conversion processes, and the neural network simulator itself. The network discovery system maintains and updates the list of available agents and their network role, which identifies the inter-modules connectivity pattern. The link management system is instantiated as soon as minimal information about the actual state of the network is gathered by the network discovery system. The link manager establishes the characteristics of data-processing connections between agents. The number, type, size, and direction of data-flows could differ in accordance to the information in the agent's genome.

The simulation starts as soon as mandatory data-processing links are established. Input/output mapping and conversion routines are executed in order to handle data translation from the internal simulator format to the format suited for data transmission on physical supports (i.e., Bluetooth, WiFi, etc.). It is possible to emulate distributed neural networks with dynamic or static topologies, with different triggering events associated with topological changes, with different synchronisation routines or different behavior patterns within given interfaces and protocols.

The Neural State Recording Facilities. The electroencephalogram is the most commonly used signal to detect and analyze brain activity. The biophysical model of EEG generation relies on the assumption that the current flows generated by clusters of simultaneously active synapses produce an elementary signal [6]. By means of virtual electrodes we aim at implementing the recordings of local field potentials of densely interconnected mesh-work of simulated neurons. The virtual electrode recorder is implemented by the simulator process and easily integrated in the neural network simulator.

3 Model Implementation

The main goal of our simulation is to emulate the biologically plausible behavior of hierarchically organized inter-connected brain areas receiving external inputs

from sensory modules and ultimately projecting to actuator modules. In addition to single unit spike trains an EEG-like signal can be recorded from each brain area. Each area is implemented by means of a neural simulator agent. Each agent is aimed to include 10,000 spiking neurons, with each neuron receiving an average of 300 synaptic inputs. Neurons and synapses may exhibit complex dynamics characterized by first order kinetics and may use combinations of arithmetic and logic functions. The design of the model is initially planned at implementing up to 64 agents in the global network of agents.

3.1 The Neural Network Simulator

The model is described elsewhere in more details [78]. Briefly, the neural network of each module is laid on a 2D lattice of neurons. At early developmental stages neural cells are differentiated into two types, excitatory (exc) and inhibitory (inh) neurons. Further differentiation mechanisms lead to the identification of input and output projecting neurons in each module. For the sake of simplicity we can consider input and output neurons forming an efferent and the afferent layers. Each such layer consists of approximately 10% of the total number of excitatory neurons. Initial connections between the populations of cells are driven by synaptogenesis process and are randomly generated according to a 2D Gaussian density function. In order to match the current Ubichip design [3] the actual modules are based on a 20×20 network which corresponds, at a mature

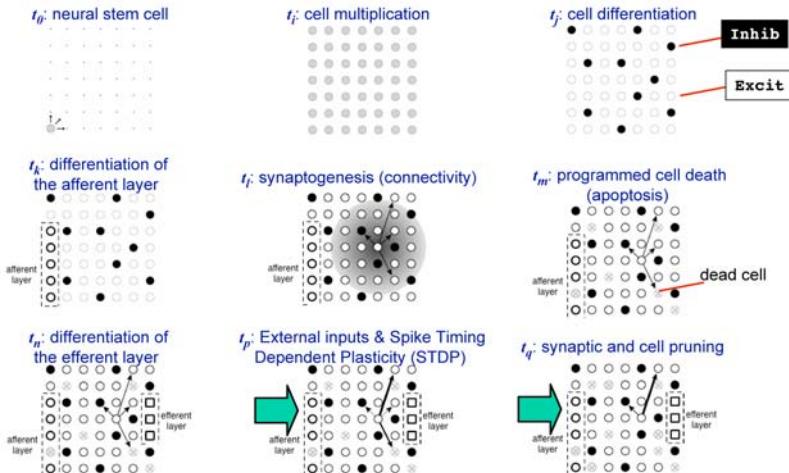


Fig. 1. Early developmental phases in the life of a neural network module. Time flows from left to right and from upper to lower panels. Grey dots represent neural stem cells, white dots excitatory neurons, black dots inhibitory neurons, crossed dots dead cells, and squares correspond to the excitatory cells that differentiated into output projecting neurons of the efferent layer.

stage of development, to 4299 ± 37 *exc-exc* (average \pm SEM), 1070 ± 15 *exc-inh*, 3918 ± 52 *inh-exc* and 961 ± 33 *inh-inh* connections.

Both types of neurons in the network are simulated by leaky integrate-and-fire neuromimes, with different tuning parameters. At each time step, the value of the membrane potential of the i -th cell $V(t)$ is calculated such that $V(t+1) = V_{\text{rest}} + \sum_j w_j(t) + Bi(t) + (1 - Si(t))((Vi(t) - V_{\text{rest}})k_{\text{mem}})$, where $w_j(t)$ is the synaptic weight from j -th to i -th neuron, V_{rest} corresponds to the value of the resting potential for the neuron, $B(t)$ is the background activity arriving to the i -th neuron, $S(t)$ is a binary state function of the i -th neuron, and k_{mem} is a membrane kinetic constant. The post-synaptic potential is implemented as a function of the relative timing of pre- and post-synaptic spikes [10], that is usually referred as spike-timing dependent plasticity (STDP). In case of synaptic depression we consider the possibility of decaying synaptic strength to zero, thus triggering synaptic pruning and ultimately cell pruning processes [11]. Fig. II illustrates the characteristic stages of early phases of each module history.

The simulator is written using the Java programming language. In addition to the proper description of the neural network it includes special routines aimed to transmit data from/to other spiking neural networks. The simulator is configured through the “genome”, which is read from the genome distributor agent in the network or from the configuration file in case of a stand-alone simulation.

3.2 Modules Distribution

We used an IP-adressing scheme to broaden the range of supported hardware platforms, reduce development time and increase overall package performance. We use the Java Agent DEvelopment [13] framework in order to work with an high-level abstract environment while developing the distributed multi-module system. The JADE platform simplifies implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications [12] and through a set of tools that support debugging and deployment phases. Thanks to its design the agent platform can be distributed across machines, which not even need to share the same OS. Like the simulator, the JADE library is fully implemented in Java, providing cross-platforms integration and allows us to focus on model development rather than on low-level system programming.

All package modules, including the neural network simulator, is packaged in JADE network agents. Then, the JADE framework is used by the hierarchical neural network stimulator routines to build up the topology of simulator agents. All inter-modules communication and data-processing are considered as message exchanges ruled by several protocols. JUBiNet reads all incoming messages from JADE message queue in a sequential manner and processes them by appropriate handlers. The processing handler sends back a new status message to the input queue as a result of the change. We use separated execution flows, in terms of processing threads, for the neural network simulator itself and for the hierarchically network logic. This allows to process network communications and simulation in parallel.

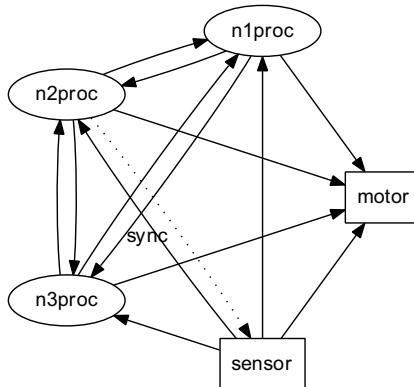


Fig. 2. Sample network topology with 1 sensory module (sensor), 3 processing modules (n1proc, n2proc, n3proc) and 1 actuator module (motor). Data-flows between agents and their directions are depicted by arrows. The dotted arrow (sync) refers to the synchronization link of the sensory module.

Inside the packages the execution flow is implemented as a sequential automata. Thread switching operations are reduced in order to increase computationally efficiency. Software testing and development is simplified by sequential data-processing routines that provide higher system stability and predictability. In accordance to this concept the network monitoring, the link manager and the input/output mapping routines are implemented as subset automates. This implementation gives us the possibility to consider a dynamic network of neural modules where some modules may enter or leave the simulation at any time. The dynamic rearrangement of the topology is handled without need of restarting all modules thanks to the the network monitoring system. This system sends notification messages to the link monitor handler about agents joined or left network and about their respective role (i.e., either sensory, actuator or processing).

Inter-Agent Data-Processing. Sensory modules are implemented by *input agents* characterized by afferences originating from sources other than other neural network simulator agents. External stimuli could be either predefined artificial stimuli or input data generated by external sensors like camera, radar, microphones, etc. In the current study we used input agents with predefined artificial spatio-temporal stimuli in order to simplify the test cycle of neural modeling and software development. Input agents have no restrictions concerning their target agents. Agents that project their activity to recipients other than the neural network simulator agents (e.g., external actuators) are called *output agents*. Output agents have no explicit limitations to their input agents. The modules that can be connected to any other neural network simulator agent are implemented by *processing agents*. An all-to-all link manager is used to establish as many inter-modules connections as allowed by each module's role. An other link manager was implemented to deal with pre-defined inter-modules topologies of special interest.

Inter-Modules Synchronization. The input/output data-processing routines used in the simulator are synchronized on a same clock cycle. The simulation actually begins only after all links are established. The execution of each next time step of the simulation is performed only after data reception is acknowledged from all known links. This method ensures simulation to run always in synchronization, but each topology modification provokes a pause in the simulation. In a static topology or in an environment with a slow rate of changes (as it happens most of the time) this implementation provides efficiency and consistency, but in a highly dynamic environment it slows the overall execution due to the large time spent in waiting status.

Input/Output Activity Mapping. The projecting pattern of an efferent neuron towards the other agents is a copy of its intra-module projecting pattern. Let us consider the example illustrated by Fig. 3. Four efferent neurons of network A, labelled $Ae1$, $Ae2$, $Ae3$ and $Ae4$, project to 1, 5, 0 and 3 cells within network A itself. This means neuron $Ae1$ will also project to 1 neuron among all possible afferent neurons of the target modules, $Ae2$ to 5 neurons among all possible afferent neurons of the target modules, and so on for all the other afferent neurons. A connectivity pattern is established based on a probabilistic basis defined by the number of potential target neurons. In this example the count of potential target neurons (i.e., the neurons belonging to the afferent layers of the target modules) is equal to 8.

This means that in case of neuron $Ae1$, each target neuron has an equiprobable chance to be connected equal to $\frac{1}{8}$. In this example the target of $Ae1$ is actually neuron $Ba1$. And so on for all other neurons. In case of a discharge pattern corresponding to cells $Ae1$ and $Ae4$ simultaneously activated the afferent neurons $Ba1$, $Ba4$, $Ca1$ and $Ca3$ would receive a postsynaptic potential. Notice that the connectivity pattern is reshuffled when topology is changed and probabilities of connection are modified.

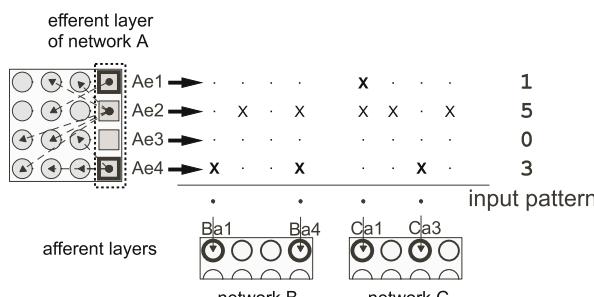


Fig. 3. Activity mapping scheme for network A connected to networks B and C. The crosses indicate the checkerboard of the active connections between network A and networks B and C. In case cells $Ae1$ and $Ae4$ are spiking, the resulting output pattern of activity is determined by the combination of the appropriate connectivity maps and provokes the excitation of cells $Ba1$, $Ba4$, $Ca1$ and $Ca3$.

3.3 Recording of Module Activity

Besides the available routines to extract spike train activity into multivariate time series, we have developed a new package designed to record the activity of a set of neurons as a function of their membrane potentials and the distance to the electrode tip, by means of virtual macro electrodes. The two main parameters of the electrode are its coordinate position $C = (x, y)$ specified by the coordinates on the neural network lattice and its *sensibility function*.

The model assumes that the electrode could be placed exactly over one neuron or exactly in-between four neighbour neurons. The sensibility function calculates the magnitude of the “electric field” generated by the neural cells at a certain distance from the electrode tip. This function is meant to combine the electrical characteristics of the electrode (mainly its impedance) and the volume conduction properties of the underlying tissue. In the simplest case it is a linear decay function, but other user-defined functions can be selected. We assume an isotropic neuropile such that for any sensibility function all neurons which are located at the same distance from the electrode tip form an equipotential layer L , thus contributing equally to the recorded signal (Fig. 4a). The electrode radius R is the total number of equipotential layers generating a recordable signal.

The equation $E_r(k) = \sum_{t=\tau \times k}^{\tau \times (k+1)-1} \sum_{r=1}^R \varphi(r) \sum_{\forall i \in L(r)} \Psi_i(t)$, calculates the electrode signal $E_r(k)$ where τ is a down-sampling parameter depending on the sampling frequency of the recording, $\varphi(r)$ is the sensibility function of the electrode, $L(r)$ is the set of all contributing neurons lying at distance r from the electrode tip located at C and forming the equipotential layer L , $\Psi_i(t)$ is an electric field function (e.g., $\Psi_i(t) = |B_i(t)| + \sum_j |w_{ji}(t)|$ or $\Psi_i(t) = V_i(t)$) depending on the model to be selected, $B_i(t)$ is the background activity afferent to the i^{th} neuron, and $w_{ji}(t)$ are the post-synaptic potentials of the j^{th} neurons projecting to the i^{th} neuron. Notice that the raw signal recorded by such virtual macro-electrode is called *electro-chipogram* (EChG) and is monopolar. Fig. 4b shows an example of such recordings, during a stimulus-driven task, with an

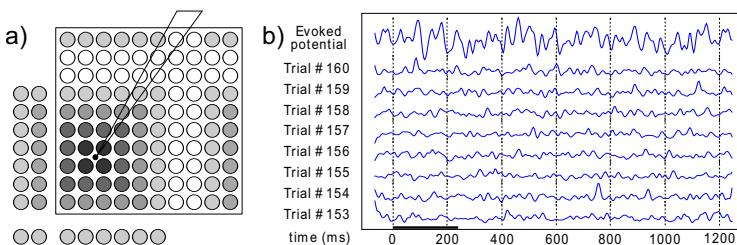


Fig. 4. a) Schematic electrode’s sensitivity area on the square neural network lattice in wrapped and unwrapped representation. Each circle represents a neuron. Intensity of black color corresponds to the intensity of the electric field recorded by the electrode. b) Local field potentials (arbitrary scale) evoked by a stimulus lasting 250 ms starting on time zero, recorded from a processing ubidule. The top trace shows the averaged signal over 8 consecutive trials. The lower traces show the individual trials.

Evoked Potential obtained by averaging a few consecutive trials. The current paper is not aimed to discuss the results of such recordings, which are now analyzed and will be extensively reported in future papers. A common reference signal generated by the spiking activity of all neurons of all modules is also recorded such to allow the generation of bipolar signals, akin of biologically recorded signals, for further analysis in a standard data format used for EEG recordings [14].

4 Discussion

We have presented a novel framework that allows the study of the activity of distributed neural networks organized in distributed interacting modules by means of virtual electrodes that record electrochipograms in each module. This framework offers a tools to study neural network interactions, complex signal processing and to compare EChG with real local field potentials and EEG recorded in experimental conditions. The current implementation of an evolving spiking neuronal model is certainly an utmost oversimplification of the reality, but the current framework opens the way to models that will embed increasingly higher biologically inspired parameters. In a separate paper we will report the first analyses of evoked EChG in a network of ubidules undergoing classical paradigms such as the odd-ball or stimulus-compatibilty tasks.

A particular feature of our approach is the possibility to enable a highly dynamic environment characterized by evolvable topologies with modules that can enter or exit the simulation at any time. The overall design is based on a highly organized system of agents messaging operated by automata, thus allowing dynamic topology “rewiring” following simple rules. The drawback is that fast changing topologies might introduce delays of information processing. This problem could be managed by implementing a separate signal recorder agent, which would receive EChG data from all simulation agents via the network. Then, the recordings could be synchronized. This solution would in turn require an increased network communication bandwidth and the appearance of a kind of centralized authority, but offer reduced agent’s file-system loads and an improvement of agent’s performance.

This new framework fits well the requirements of sophisticated control circuits for robotic implementations in collective behavioral studies where each robot is driven by one or more neural networks (*e.g.* [15]). Mobile version of JADE compatible framework [13] could be used in order to obtain a platform with reduced footprint and compatibility with mobile Java environments.

Acknowledgments. The authors acknowledge the support by the European Union FP6 grant #034632 (PERPLEXUS) and the contributions of J. Iglesias for the simulator and of O. Brousse, Th. Gil, G. Sassatelli and F. Grize for the JADE integration.

References

1. Haikonen, P.O.: Robot Brains: Circuits and Systems for Conscious Machines. WileyBlackwell (2007)
2. Sanchez, E., Perez-Uribe, A., Upegui, A., Thoma, Y., Moreno, J.M., Villa, A., Volken, H., Napieralski, A., Sassatelli, G., Lavarec, E.: Perplexus: Pervasive computing framework for modeling complex virtually-unbounded systems. In: AHS 2007: Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems, Washington, DC, USA, pp. 587–591. IEEE Computer Society, Los Alamitos (2007)
3. Upegui, A., Thoma, Y., Sanchez, E., Perez-Uribe, A., Moreno, J.M., Madrenas, J., Sassatelli, G.: The perplexus bio-inspired hardware platform: A flexible and modular approach. *Int. J. Know.-Based Intell. Eng. Syst.* 12(3), 201–212 (2008)
4. Brousse, O., Sassatelli, G., Gil, T., Robert, M., Grize, F., Sanchez, E., Upegui, A., Thoma, Y.: The perplexus programming framework: Combining bio-inspiration and agent-oriented programming for the simulation of large scale complex systems. In: Hornby, G.S., Sekanina, L., Haddow, P.C. (eds.) ICES 2008. LNCS, vol. 5216, pp. 402–407. Springer, Heidelberg (2008)
5. Benoskova, L., Kasabov, N.: Computational Neurogenetic Modeling. Springer, New York (2007)
6. Elul, R.: The genesis of the eeg. *Int. Rev. Neurobiol.* 15, 227–272 (1972)
7. Iglesias, J., Villa, A.E.P.: Effect of stimulus-driven pruning on the detection of spatiotemporal patterns of activity in large neural networks. *BioSystems* 89, 287–293 (2007)
8. Iglesias, J., Chibirova, O., Villa, A.: Nonlinear dynamics emerging in large scale neural networks with ontogenetic and epigenetic processes. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 579–588. Springer, Heidelberg (2007)
9. Roberts, P.D., Bell, C.C.: Spike timing dependent synaptic plasticity in biological systems. *Biol. Cybern.* 87, 392–403 (2002)
10. Izhikevich, E.M., Gally, J.A., Edelman, G.M.: Spike-timing dynamics of neuronal groups. *Cerebral Cortex* 14, 933–944 (2004)
11. Iglesias, J., Eriksson, J., Grize, F., Tomassini, M., Villa, A.E.: Dynamics of pruning in simulated large-scale spiking neural networks. *BioSystems* 79(1), 11–20 (2005)
12. FIPA TC C: Fipa communicative act library specification. Technical report, IEEE Foundation for Intelligent Physical Agents (2001)
13. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems With Jade. Wiley, Wiltshire (2007)
14. Kemp, B., Olivan, J.: European data format [‘plus’] (edf+), an edf alike standard format for the exchange of physiological data. *Clinical Neurophysiology* 114(9), 1755–1761 (2003)
15. Dorigo, M., Trianni, V., Sahin, E., Gros, R., Labella, T.H., Baldassarre, G., Nolfi, S., Mondada, F., Deneubourg, J.L., Floreano, D., Gambardella, L.M.: Evolving Self-Organizing Behaviors for a Swarm-bot. *Autonomous Robots, special Issue on Swarm Robotics* 17(2-3), 223–245 (2004); September - November 2004 Sponsor: swarm-bots, OFES 01-0012-1

Continuous Attractors of Lotka-Volterra Recurrent Neural Networks

Haixian Zhang¹, Jiali Yu¹, and Zhang Yi²

¹ School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, P.R. China
{hxzhang,yujiali}@uestc.edu.cn

² Machine Intelligence Laboratory, School of Computer Science, Sichuan University, Chengdu 610065, P.R. China
zhangyi@scu.edu.cn

Abstract. Continuous attractor neural network (CANN) models have been studied in conjunction with many diverse brain functions including local cortical processing, working memory, and spatial representation. There is good evidence for continuous stimuli, such as orientation, moving direction, and the spatial location of objects could be encoded as continuous attractors in neural networks. Although their wide applications for the information processing in the brain, representation and stability analysis of continuous attractors in non-linear recurrent neural networks (RNNs) have been reported very little so far. This paper studies the continuous attractors of Lotka-Volterra (LV) recurrent neural networks. Conditions are given to insure the network has continuous attractors. Representation of continuous attractor is obtained under the conditions. Simulations are employed to illustrate the theory.

Keywords: Continuous attractors, Recurrent neural networks, Convergence.

1 Introduction

Continuous attractor neural networks (CANNs) have received wide attention by it's good property in caving the landscape of the cognitive functions, such as movement control, spatial navigation, population decoding and object categorization, see for examples, [1], [2], [17], [6], [10], [14], [12]. There are strong indications that continuous attractors and dynamical mechanisms are used frequently for information processing in the brain. In some neurobiological models, continuous attractors have been used to represent continuous quantities like working memory in prefrontal cortex, orientation of a visual stimulus, eye position, head direction, and so on, see for examples, [5] [17], [6], [11], [13].

Continuous attractors neural network models have been studied by many authors, see for examples, [7], [8], [9], [3], [2], [15], [16]. These studies have illustrated clearly that CANNs can track moving continuous stimulus . Despite these success, however, detailed rigorous analysis of the presentation of continuous attractors in a CANN are still lacking. These include, such as, the condition under

that a CANN can have continuous attractors; the presentation of a continuous attractor to demonstrate what a continuous attractor is.

The theory of line attractors of linear recurrent neural networks (RNNs) has been used successfully to explain how the brain can keep the eyes still in [6]. More general cases of continuous attractors has been studied for linear RNNs in [4], it gives the explicit representation of the continuous attractors in linear RNNs.

The continuous attractors of a class of recurrent neural networks with nonlinear transform function called linear-threshold(LT) is also studied in [4]. However, general results for continuous attractors in nonlinear RNNs have been reported very little so far. This paper studies the continuous attractors of a class of nonlinear RNNs, Lotka-Volterra (LV) recurrent neural networks. Conditions are given to insure the network has continuous attractors. The representation of continuous attractor is also given to show what a continuous attractor of this kind of network is. The continuous attractors can be completely described by the representations.

The paper is organized as follows. Preliminaries are given in Section 2. Continuous attractors of Lotka-Volterra recurrent neural networks are given in Section 3. Simulations are given in Section 4 to illustrate the theories. Finally, conclusion is given in Section 5.

2 Preliminaries

The model of Lotka-Volterra recurrent neural networks is described by

$$\dot{x}_i(t) = x_i(t) \cdot \left[-x_i(t) + \sum_{j=1}^n w_{ij} x_j(t) + b_i \right] \quad (1)$$

for $t \geq 0$, where x_i denotes the activity of neuron i and $x = (x_1, x_2, \dots, x_n)^T \geq 0$ denotes the state of the network, $W = (w_{ij})_{n \times n}$ is real symmetric $n \times n$ matrix, each of their elements represents the synaptic connection weight of neuron j to neuron i , $b = (b_1, b_2, \dots, b_n)^T \in R^n$ denotes the external inputs. The network can also be written as vector form

$$\dot{x}(t) = diag[x(t)] \cdot [-x(t) + Wx(t) + b] \quad (2)$$

where $diag[x(t)]$ is a $n \times n$ matrix with the main diagonal elements are $x_i(t)$ ($i = 1, 2, \dots, n$), the other elements are all zero.

The concept of Lyapunov stability is associated with some equilibrium points. An equilibrium point can be stable or unstable. However, in practice, only the outputs of stable equilibrium points can be observed. Next, the definition of equilibrium point and Lyapunov stability for an equilibrium point are given.

Definition 1. A point of x^* is called an equilibrium point of the network (1), if it holds that

$$x_i^* \left[-x_i^* + \sum_{j=1}^n w_{ij} x_j^* + b_i \right] = 0, \quad (i = 1, \dots, n). \quad (3)$$

Definition 2. An equilibrium point x^* is called stable, if given any constant $\epsilon > 0$, there exists a constant $\delta > 0$ such that

$$\|x(0) - x^*\| \leq \delta$$

implies that

$$\|x(t) - x^*\| \leq \epsilon$$

for all $t \geq 0$.

Next, the definition of continuous attractors will be given.

Definition 3. A set of equilibrium points C is called a continuous attractor if it is connected set and each point $x^* \in C$ is stable.

Definition 4. Let M be an $n \times n$ matrix, and Let $P \subseteq \{1, 2, \dots, n\}$ be an index set. The matrix M_P is said to be a submatrix of M if the matrix M_P can be constructed from M simply by removing from M all rows and columns not indexed by P .

Given two index sets $P, Z \subseteq \{1, 2, \dots, n\}$ and $P \cup Z = \{1, 2, \dots, n\}$. The network (2) can be rewritten as

$$\begin{cases} \dot{x}_P(t) = \text{diag}[x_P(t)] \cdot [-x_P(t) + W_P \cdot x_P(t) + W_{PZ} \cdot x_Z(t) + b_P] \\ \dot{x}_Z(t) = \text{diag}[x_Z(t)] \cdot [-x_Z(t) + W_{ZP} \cdot x_P(t) + W_Z \cdot x_Z(t) + b_Z] \end{cases} \quad (4)$$

for $t \geq 0$, where W_P and W_Z are submatrix of W . W_{PZ} is a matrix constructed from W by removing all rows not indexed by P and all columns not indexed by Z , and matrix W_{ZP} are constructed by the same way.

Next, some useful lemmas will be given.

Lemma 1. Given any $x_i(0) \geq 0$, it holds that

$$x_i(t) \begin{cases} > 0, & \text{if } x_i(0) > 0 \\ = 0, & \text{if } x_i(0) = 0 \end{cases}$$

for all $t \geq 0$ and $i = 1, \dots, n$.

Proof: Denote

$$r_i(t) = -x_i + \sum_{j=1}^n w_{ij} x_j + b_i$$

for $t \geq 0$ and $i = 1, 2, \dots, n$. Then, it follows that

$$x_i(t) = x_i(0) \cdot e^{\int_0^t r_i(s) ds} \quad (5)$$

for $t \geq 0$.

From (5), it follows that $x_i(t) > 0$ if the initial condition $x_i(0) > 0$, and $x_i(t) = 0$ if $x_i(0) = 0$, ($i = 1, 2, \dots, n$) for all $t \geq 0$. This completes the proof.

Lemma 2. *Let x^* be an equilibrium point. Then, the linearization of the network (7) at x^* is given by:*

$$\begin{aligned} \frac{d[x_i(t) - x_i^*]}{dt} &= [x_i(t) - x_i^*][-x_i^* + \sum_{j=1}^n w_{ij}x_j^* + b_i] \\ &\quad x_i^*[-(x_i(t) - x_i^*) + \sum_{j=1}^n w_{ij}(x_j(t) - x_j^*)] \end{aligned} \quad (6)$$

for $t \geq 0$ and $1 \leq i \leq n$.

Proof: From (II), it follows that

$$\begin{aligned} \frac{d[x_i(t) - x_i^*]}{dt} &= [x_i(t) - x_i^*][-x_i^* + \sum_{j=1}^n w_{ij}x_j^* + b_i] \\ &\quad x_i^*[-1 + w_{ii}][x_i(t) - x_i^*] + x_i^*[\sum_{j \neq i} w_{ij}(x_j(t) - x_j^*)] + o(x_i(t) - x_i^*) \\ &= [x_i(t) - x_i^*][-x_i^* + \sum_{j=1}^n w_{ij}x_j^* + b_i] \\ &\quad x_i^*[-(x_i(t) - x_i^*) + \sum_{j=1}^n w_{ij}(x_j(t) - x_j^*)] + o(x_i(t) - x_i^*) \end{aligned}$$

for $t \geq 0$. The result follows by removing the higher order term. This completes the proof.

3 Continuous Attractors of LV Neural Networks

Since matrix W is symmetric and W_P is a submatrix of W , it is easy to check that W_P is also symmetric. Let λ_i^P ($i = 1, 2, \dots, p$) be all the eigenvalues of W_P ordered by $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$. S_i ($i = 1, 2, \dots, p$) is the eigenvector of W_P corresponding to the eigenvalue λ_i^P . Suppose the multiplicity of λ_1^P is m , denote $V_{\lambda_1^P}^P = \sum_{i=1}^m c_i S_i$, ($c_i \in R$) be the eigen-subspace associated with the eigenvalue λ_1^P . Suppose that

$$b_P = \sum_{i=1}^p \tilde{b}_i^P \cdot S_i^P.$$

Theorem 1. *Suppose that $\lambda_1^P = 1$ and $b_P \perp V_{\lambda_1^P}^P$. If it holds that*

$$\begin{cases} \sum_{i=1}^m c_i S_i^P + \sum_{j=m+1}^p \frac{\tilde{b}_j^P}{1-\lambda_j^P} S_j^P > 0, \\ W_{ZP} \cdot \left[\sum_{i=1}^m c_i \cdot S_i^P + \sum_{j=m+1}^p \frac{\tilde{b}_j^P}{1-\lambda_j^P} S_j^P \right] + b_Z < 0 \end{cases} \quad (7)$$

for any constants $c_i \in R(i = 1, \dots, m)$, Then,

$$C = \left\{ \left[\sum_{i=1}^m c_i S_i^P + \sum_{j=m+1}^p \frac{\tilde{b}_j^P}{1-\lambda_j^P} S_j^P \right] \middle| c_i \in R(i = 1, \dots, m) \right\} \quad (8)$$

is a continuous attractor of the network (7).

Proof: Clearly, C is a connected set. Since $b_P \perp V_{\lambda_1^P}$, then $\tilde{b}_1^P = \dots = \tilde{b}_m^P = 0$. Given any $x^* \in C$, there exist constant $c_i \in R(i = 1, \dots, m)$ such that

$$\begin{cases} x_P^* = \sum_{i=1}^m c_i S_i^P + \sum_{j=m+1}^p \frac{\tilde{b}_j^P}{1-\lambda_j^P} S_j^P > 0, \\ x_Z^* = 0. \end{cases}$$

It is easy to check that x^* is an equilibrium point of the network (1).

Next, we will show that x^* is stable.

Denote

$$\alpha \triangleq \max_{i \in P} \{x_i^*\}, \beta \triangleq \min_{i \in P} \{x_i^*\},$$

and

$$\gamma \triangleq \min_{i \in Z} \left\{ - \left(\sum_{j=1}^p w_{ij} x_j^* + b_i \right) \right\}.$$

Given a constant $\epsilon > 0$ such that

$$0 < \epsilon \leq \min \{x_i^*, i = 1, 2, \dots, p\},$$

denote

$$M = \min \left\{ \frac{\sqrt{\beta/\alpha}}{2}, \frac{\gamma}{2\alpha \|W_{PZ}\|} \right\},$$

and $\delta = M\epsilon$, define a neighborhood B_δ of $x(0)$ by

$$B_\delta = \{ |x_i(0) - x_i^*| \leq \delta, i = 1, 2, \dots, n\}.$$

Given $i \in Z$, from lemma 2 and Eq.(4), it follows that

$$\frac{dx_i(t)}{dt} = \left[\sum_{j=1}^p w_{ij} x_j^* + b_i \right] x_i(t) \quad (9)$$

for $t \geq 0$. That is

$$x_i(t) = x_i(0) \cdot \exp \left(\left[\sum_{j=1}^p w_{ij} x_j^* + b_i \right] t \right)$$

for $t \geq 0$. By (7), clearly, $\gamma > 0$. Then,

$$\| x_Z(t) \| \leq \| x_Z(0) \| \cdot e^{-\gamma t} \leq \delta < \epsilon \quad (10)$$

for $t \geq 0$.

Given $i \in P$, it follows that

$$\frac{d[x_i(t) - x_i^*]}{dt} = \text{diag}(x_i^*) \left[-(x_i(t) - x_i^*) + \sum_{j=1}^n w_{ij}(x_j(t) - x_j^*) \right] \quad (11)$$

for $t \geq 0$.

Define the following function

$$V(t) = \frac{1}{2} \sum_{i \in P} \frac{[x_i(t) - x_i^*]^2}{x_i^*}$$

for $t \geq 0$. Clearly,

$$\frac{1}{2\alpha} \| x_P(t) - x_P^* \|^2 \leq V(t) \leq \frac{1}{2\beta} \| x_P(t) - x_P^* \|^2$$

for $t \geq 0$.

From Eq.(II), it gives that

$$\begin{aligned} \dot{V}(t) &= \frac{1}{2} \sum_{i \in P} \frac{2[x_i(t) - x_i^*]}{x_i^*} \cdot \frac{d[x_i(t) - x_i^*]}{dt} \\ &= (x_P(t) - x_P^*)^T \cdot [(W_P - I_P)(x_P(t) - x_P^*) + W_{PZ}x_Z(t)] \\ &= (x_P(t) - x_P^*)^T (W_P - I_P)(x_P(t) - x_P^*) + (x_P(t) - x_P^*)^T W_{PZ}x_Z(t) \\ &\leq \| (x_P(t) - x_P^*) \| \cdot \| W_{PZ} \| \cdot \| x_Z(t) \| \\ &\leq \sqrt{2\alpha} \cdot \| W_{PZ} \| \| x_Z(0) \| \cdot e^{-\gamma t} \sqrt{V(t)} \end{aligned}$$

for $t \geq 0$. Then,

$$\sqrt{V(t)} \leq \sqrt{V(0)} + \frac{\sqrt{\alpha} \cdot \| W_{PZ} \| \cdot \| x_Z(0) \|}{\sqrt{2\gamma}} \cdot (1 - e^{-\gamma t})$$

for $t \geq 0$. Thus,

$$\begin{aligned} \| x_P(t) - x_P^* \| &\leq \sqrt{2\alpha} \cdot \left(\sqrt{V(0)} + \frac{\sqrt{\alpha} \cdot \| W_{PZ} \| \cdot \| x_Z(0) \|}{\sqrt{2\gamma}} \right) \\ &\leq \sqrt{\frac{\alpha}{\beta}} \cdot \| x_P(0) - x_P^* \| + \frac{\alpha \| W_{PZ} \|}{\gamma} \cdot \| x_Z(0) \| \\ &\leq \epsilon \end{aligned} \quad (12)$$

for $t \geq 0$.

From Eq.(10) and (12), it follows that

$$\|x(t) - x^*\| \leq \epsilon$$

for $t \geq 0$. By definition 2, x^* is a stable equilibrium point of the network (11). Thus, Eq.(8) is a continuous attractor of the network. This completes the proof.

4 Simulations

In this section, we will give some simulations to illustrate the continuous attractors theory established in above section.

Example 1: Let us first consider a two dimensional Lotka-volterra recurrent neural networks.

$$\dot{x}(t) = \text{diag}[x(t)] \cdot \left[-x(t) + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} x(t) \right] \quad (13)$$

for $t \geq 0$. We denote

$$W = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

It can be checked that the largest eigenvalue of W is 1 with multiplicity 1. By Theorem 1, the network exists a line continuous attractor. Figure 1 shows the continuous attractors of the network. The figure shows that forty trajectories starting from randomly selected initial points converge to the line attractor.

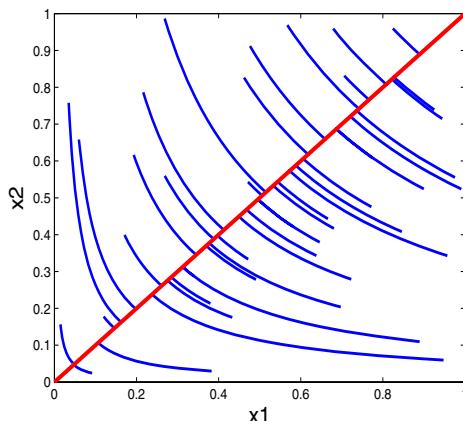


Fig. 1. Continuous attractors of Lotka-Volterra recurrent neural network in Example 1. The thick straight line is the continuous attractors. It is a connected set composed of stable equilibrium points of the network (13).

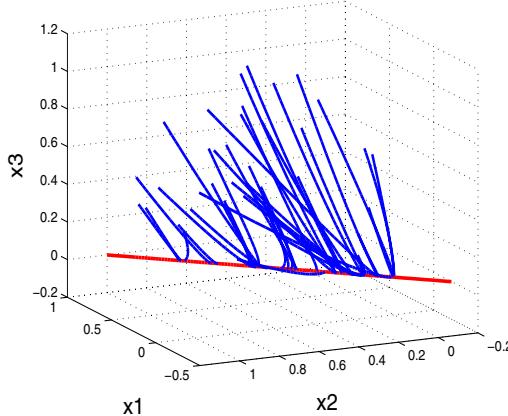


Fig. 2. Continuous attractors of Lotka-Volterra recurrent neural network in Example 2. The thick straight line is the continuous attractors. It is lies in the plane of $x_3 = 0$.

Example 2: Let us consider a three dimensional Lotka-Volterra recurrent neural networks.

$$\dot{x}(t) = \text{diag}[x(t)] \cdot \left[-x(t) + \begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & -2 \\ -2 & -2 & 0 \end{bmatrix} x(t) \right] \quad (14)$$

for $t \geq 0$. Denote

$$W = \begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & -2 \\ -2 & -2 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Let $P = \{1, 2\}$, then

$$W_P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad W_{ZP} = [-2, -2].$$

It can be easily checked that the eigenvalues of W_P are 1 and -1 . $S_1^P = [1, 1]$ is the eigenvector corresponding to eigenvalue 1, denote V_1^P is the eigen-subspace developed by S_1^P . Clearly, $b_P \perp V_1^P$. By Theorem II, the network has a continuous attractor with the state value of the third neuron is zero. Figure 2 shows that forty trajectories starting from randomly selected initial points converge to the line attractor, and this line attractor lies in the plane of $x_3 = 0$.

5 Conclusion

In this paper, continuous attractors of Lotka-Volterra recurrent neural networks are studied. Conditions to insure the network has continuous attractors are successfully obtained. Explicit representation of continuous attractor is also obtained under the given conditions. Simulations well verify the theory.

Acknowledgment

This work was supported by Chinese 863 High-Tech Program under Grant 2007AA01Z321.

References

1. Amari, S.: Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics* 27, 77–87 (1977)
2. Ben-Yishai, R., Bar-Or, R.L., Sompolinsky, H.: Theory of orientation tuning in visual cortex. *Proc. Nat. Acad. Sci. USA* 92, 3844–3848 (1995)
3. Compte, A., Brunel, N., Goldman-Rakic, P.S., Wang, X.J.: Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cereb. Cortex* 10, 910–923 (2000)
4. Yu, J., Zhang, Y., Zhang, L.: Representation of continuous attractors of recurrent neural networks. *IEEE Transactions on Neural Networks* 20(2), 368–372 (2009)
5. Salinas, E.: Background synaptic activity as a switch between dynamical states in network. *Neural computation* 15, 1439–1475 (2003)
6. Seung, H.S.: How the brain keeps the eyes still. *J. Neurobiology* 93, 13339–13344 (1996)
7. Seung, H.S.: Continuous attractors and oculomotor control. *Neural Networks* 11, 1253–1258 (1998)
8. Seung, H.S.: Learning continuous attractors in recurrent networks. *Adv. Neural Info. Proc. Syst.* 10, 654–660 (1998)
9. Seung, H.S., Lee, D.D.: The manifold ways of perception. *Science* 290, 2268–2269 (2000)
10. Samsonovich, A., McNaughton, B.L.: Path integration and cognitive mapping in a continuous attractor neural network model. *J. Neurosci.* 7, 5900 (1997)
11. Stringer, S.M., Trappenberg, T.P., Rolls, E., Aranjo, I.: Self organizing continuous attractor networks and path integration: One-dimensional models of head direction cell. *Network: Computation in Neural Systems* 13, 217–242 (2002)
12. Trappenberg, T.P., Standage, D.I.: Self-organising continuous attractor networks with multiple activity packets, and the representation of space. *Neural Networks* 17, 5–27 (2004)
13. Tsodyks, M., Sejnowski, T.: Associative memory and hippocampal place cells. *International journal of neural systems* 6, 81–86 (1995)
14. Wu, S., Amari, S., Nakahara, H.: Population coding and decoding in a neural fields: a computational study. *Neural Computation* 14, 999–1026 (2002)
15. Wu, S.: Shun-Ichi Amari, Computing with continuous attractors: Stability and online aspects. *Neural Computation* 17, 2215–2239 (2005)
16. Wu, S., Hamaguchi, K., Amari, S.: Dynamics and computation of continuous attractors. *Neural Computation* 20, 994–1025 (2008)
17. Zhang, K.C.: Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A Theory. *The journal of neuroscience* 16(6), 2110–2126 (1996)

Learning Complex Population-Coded Sequences

Kiran V. Byadarhaly, Mithun Perdoor, Suresh Vasa, Emmanuel Fernandez,
and Ali A. Minai*

Department of Electrical & Computer Engineering, University of Cincinnati,
Cincinnati, OH 45221-0030

bkirav@yahoo.com, perdoomc@email.uc.edu, vasash@email.uc.edu,
emmanuel.fernandez@uc.edu, ali.minai@uc.edu

Abstract. In humans and primates, the sequential structure of complex actions is apparently learned at an abstract “cognitive” level in several regions of the frontal cortex, independent of the control of the immediate effectors by the motor system. At this level, actions are represented in terms of kinematic parameters – especially direction of end effector movement – and encoded using population codes. Muscle force signals are generated from this representation by downstream systems in the motor cortex and the spinal cord.

In this paper, we consider the problem of learning population-coded kinematic sequences in an abstract neural network model of the medial frontal cortex. For concreteness, the sequences are represented as line drawings in a two-dimensional workspace. Learning such sequences presents several challenges because of the internal complexity of the individual sequences and extensive overlap between sequences. We show that, by using a simple module-selection mechanism, our model is capable of learning multiple sequences with complex structure and very high cross-sequence similarity.

Keywords: Sequence learning, population coding, motor system.

1 Introduction

The ability to act in complex and purposive ways is a central attribute of higher animals, and is essential to the understanding of intelligence and cognition. Over the last several decades, neuroscientists and cognitive scientists have elucidated many of the mechanisms underlying motor control. Experiments with primates have shown that complex voluntary movements are encoded at multiple levels in the cortex, brainstem and the spinal cord [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]. However, the overall process by which such movements are generated remains a subject of significant debate [16,17,18,19,20].

Research with simple animals such as lampreys and salamanders [21,22] has suggested the existence of *motor programs* – pre-configured patterns of movements that can be triggered as a whole by selection through the basal ganglia

* Corresponding author.

and the brainstem. This view has often been applied to the higher vertebrates as well, and offers a plausible explanation for simple movements such as walking, swallowing, reaching, etc. However, much more complex movements – such as writing, speaking, playing on a keyboard, pitching a baseball – can become integrated into a precisely repeatable, yet flexible motor program, and recalled as a *single entity* when needed. This “chunking” [23] is thought to arise through a combination of pattern recognition and sequence learning, and is the focus of this paper.

2 Background and Motivation

All complex movements can be considered sequences of simpler movements, and there is considerable experimental evidence showing that neural activity explicitly encodes the sequential structure of movement at several levels [1][2][4][19]. However, in many cases – e.g., writing, speaking, playing on a keyboard, etc. – what might be learned initially as a sequence of simple movements eventually becomes a single complex movement. For example, a child may learn to write letters by connecting dots, or as sequences of pencil strokes, but eventually, each of these sequences becomes stored as a *single object* – still sequential in structure, but triggered by a single cognitive command, e.g., “write A”. Something similar must happen as the child learns to write whole words without spelling them out, or speaking them as continuous phoneme sequences. Sequential tasks are also used extensively as an experimental paradigm by researchers studying the motor system [3][4][9][10][11][12][14][19]. Based on these considerations, sequence learning has been considered a fundamental issue in motor control, and several models have been developed for it [24][25][26][27][28][29][30][31]. However, such sequence learning still presents several challenges.

Though partially challenged by some recent data [14][19], the standard view of motor control [6] postulates two distinct system levels. The higher level system, comprising the prefrontal cortex (PFC), the premotor cortex (PM), the supplementary motor area (SMA) and the pre-SMA, encodes kinematic information such as direction and velocity as well as information about task context [3][4][9][8][10][11][12][19]. In particular, the direction of end effector movement is represented using *population codes* [12] defined over neurons tuned to particular *preferred directions* (see next section for details). In contrast, downstream systems including the motor cortex (M1), the brainstem and the spinal cord (SC) increasingly encode posture-dependent muscle force signals needed to execute the actions [16][17][20]. Thus, with reference to sequential actions, the overall system implements a convenient division of labor, where motor sequences such as writing letters or words, drawing shapes, playing key sequences, etc., are learned as *action programs* at an abstract, posture- and effector-independent kinematic level in the PFC/PM/SMA system, and are then “translated” into sequences of motor commands by M1 and SC [3], possibly guided by the basal ganglia and cerebellum [23][32][29][33]. Of course, this does not preclude the possibility that sequence recognition and learning also occurs in M1 [5][14][19], and the model we

study here could well apply to population-coded representations in M1 as well. However, we consider purely kinematic action programs, encoded as sequences of direction population codes.

Several difficulties are inherent in computational models for learning population-coded action sequences, including the following: 1) Successive elements of an individual sequence may be very similar or even identical (e.g., representing several moves in the same direction); 2) Within an individual sequence, the same element may be followed by different successors at different points, requiring disambiguation [34]; 3) Different sequences may overlap strongly, e.g., the sequences for writing an O and a C may share most of their structure; 4) The directional information encoded in the population code may be noisy over different learning trials. These features mean that a simple heteroassociative scheme linking successive elements is not feasible. Approaches have been suggested to address some of these problems for non-population coded [35,36,34,27,28,37], and population-coded [24,25,26] data.

In this paper, we present an abstract but biologically motivated model of how population-coded action sequences may be learned in the brain. This model is informed by experimental studies on many areas of the frontal cortex during sequence encoding and learning tasks. Some neurons in the prefrontal cortex fire preferentially at the beginning of action sequences [38], while others remain active for the duration of the sequence [39]. These signals go to both the SMA/PM region, and to the basal ganglia and cerebellum, where they are hypothesized to select [23,29,38,33] and pace [31,33] action sequences, respectively. Neurons with sequence-specific activity, i.e., sequence identifiers, have been shown in pre-SMA, while neurons encoding element order in action sequences have been found in both PFC and SMA [3,40,9,7,10,38]. Our model incorporates these observations to build upon an earlier sequence learning model [27]. In particular, the system can learn highly similar sequences by sequence-specific selection of neural groups within the network. This is an example of switched modularity, which we have proposed as a general mechanism for information processing in complex biological networks [41,42,43,44]. Computational models of such switching mediated by the basal ganglia have been proposed [29,33]. The system is studied using a drawing/writing task in a 2-D workspace.

3 Model Description

A complex drawing, e.g., a letter or shape, is modeled as a sequence of *elementary actions* starting from a canonical initial point. Each elementary action results in the drawing of a short segment of fixed length in a specific direction on the workspace. This is similar to tasks used in behavioral experiments with primates [3,4,9,10,11,12,14,19]. Thus, for example, an L-shape starting at the top is a sequence of downward segments followed by a sequence of leftward ones.

The system learns action sequences from noisy exemplars, where each elementary action – henceforth called *action* – is represented as a population code over a set of directionally tuned neurons. We assume that this representation arises in cortical regions receiving sensory input, and is available during learning.

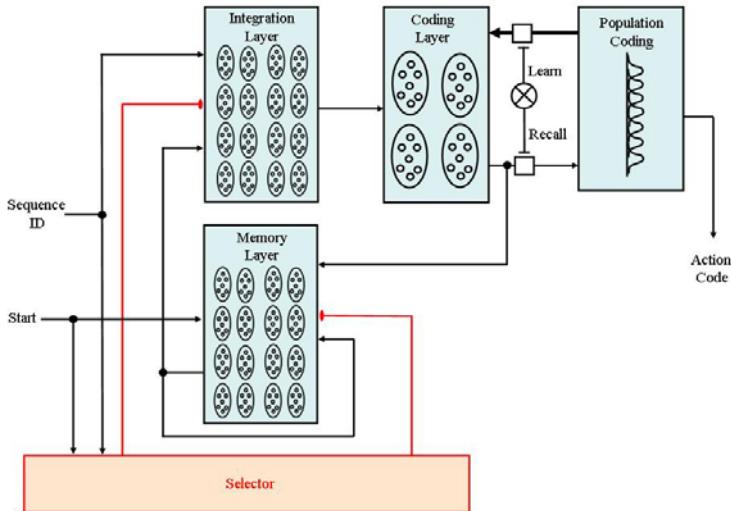


Fig. 1. The sequence learning module

The structure of the sequence learning system is shown in Figure 1. For learning, the system receives three inputs: 1) A *start* input, marking the beginning of a new sequence (corresponding to the PFC signal found in [38]); 2) An *action* input, which is the population code of the current elementary action (corresponding to representations found in PFC [10] and SMA [3,40,97]); and 3) A *sequence ID* input, which represents the identity of the action sequence (corresponding to sequence-specific activity in the pre-SMA [40,97]).

The system operates in two modes: 1) A *learning mode*, in which it receives sequences of action inputs and learns them in association with the given sequence ID; and 2) A *recall mode*, in which it receives no action input and recalls the sequence associated with the given sequence ID. The recalled sequence is decoded into the original population code by the population coding system. This and other parts of the system are described below in greater detail.

3.1 Population Coding

Each action in the system is encoded by its direction using a population code [1]. This is done by a layer of N_P directionally-tuned neurons. The response of a neuron, k , in this layer is given by a cosine function [24]: $z_k = \cos(\theta_d - \theta_k^*)$, where θ_d is the currently coded direction (represented as an angle in a fixed reference frame), and θ_k^* is the preferred direction of neuron k (note that z_k can be rectified by adding an offset, but we do not use that here for simplicity.) The preferred directions for the neurons in the layer are chosen to cover 0° to 360° uniformly, so the vector, $z^P = [z_1 \dots z_{N_P}]$ is a population coded representation of the direction θ_d — and thus of the corresponding action. The direction can be recovered from it by using a simple decoding function [4]: $\hat{\theta} = \text{phase} \left[\frac{1}{N_P} \sum_{k=1}^{N_P} z_k \exp(i\theta_k^*) \right]$.

The direction input to the population coder comes from the sensory areas (not simulated) during learning and from the sequence learning system during recall.

3.2 Sequence Learning System

The sequence learning system learns abstract representations of action sequences from noisy exemplars. As shown in Figure 1, the core of the system comprises three layers, each implemented as a set of winner-take-all (WTA) clusters. These clusters may be considered an abstraction of cortical hypercolumns [45]. Activity in this core region – corresponding roughly to parts of the PFC and SMA – is selectively biased in a sequence-specific way by the *selector* system, which plays the role hypothesized for the basal ganglia [23,39,32,29,30,38,33].

The *coding layer*, C , has N_C neurons organized into M_C clusters of $n_C = N_C/M_C$ neurons each. Activity within each cluster is WTA, so only the most strongly activated neuron in the cluster can fire. Non-selected clusters have no activity. The *memory layer*, R , is a recurrently connected layer of N_R neurons, organized into M_R clusters of m_r neurons each. Only $m_R \leq M_R$ clusters are selectively activated at a time by the signal from the selector system, and activity in each selected cluster is WTA. The *integration layer*, I , has N_I neurons clustered into M_I WTA clusters of n_I neurons each. Of these, $m_I \leq M_I$ clusters are selectively activated at a time by the selector system. The selector system, B , is a heteroassociative network that maps each input into a particular sparse pattern of $N_B = M_R + M_I$ binary outputs, each of which gates a cluster in the memory or integration layers. This architecture is a simplified version of that found in the BG, whose neurons target very specific patches of the cortex [23,32].

During learning, the coding layer receives a very strong action input, $X(t)$, from the population coder, generating a sparse representation, $Z^C(t)$, of the current observed action. This is projected to the memory layer via fixed random weights. The memory layer also receives input from itself through fixed random weights, so that its current state, $Z^R(t)$, comes to represent a compressed code for the activity of the coding layer over the last several steps. The integration layer receives the current sequence ID input as well as $Z^R(t)$ through fixed random weights, and integrates them into a sequence-specific representation, $Z^I(t)$, of the sequence up to that point. This representation is then associated with the next coding layer state, $Z^C(t+1)$, through modifiable weights from Layer I to Layer C using a delta learning rule [46]. In both the R and I layers, the selector system selects sequence-specific groups of clusters, thus ensuring that representations for differently identified sequences remain distinct even if the sequences are similar. The *start* signal is input to the memory layer, where it triggers a heteroassociatively stored *initial representation*, thus ensuring that the integration layer always receives the same input at the start of a sequence. The *start* signal also resets the selector system.

During recall, the flow between the population coder and the coding layer is reversed, and the system is given only the sequence ID and *start* inputs. This then triggers the recall of the previously learned sequence as the coding and memory

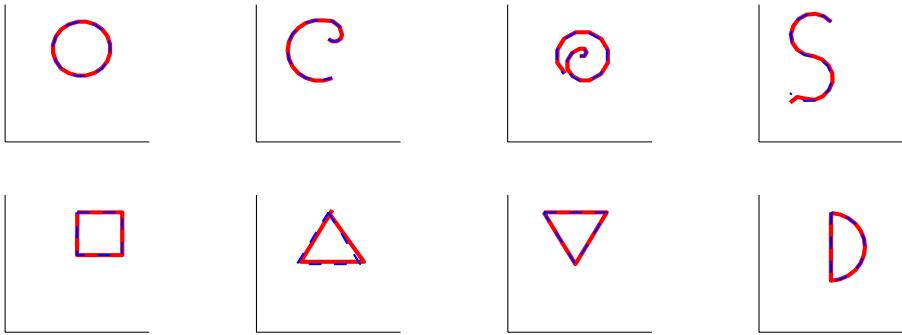


Fig. 2. Recall of eight learned figures. The dashed (blue) lines show the ideal figures, the solid (red) lines the actual figures generated after learning. Ideal and actual figures coincide in some plots. The input has a small amount of angular noise.

layers recover their states during learning. The activity of Layer C is decoded by the population coding system into the population code for the corresponding actions. This decoding is based on training of the weights from Layer C to the population coder using a delta rule [46]. This learning can occur concurrently with the sequence learning or separately via ideomotor babbling.

4 Simulations and Results

An implementation of the system with $M_C = 30$, $M_R = 60$, $M_I = 60$ and $n_C = n_R = n_I = n$ was simulated using a canonical repertoire of 8 shapes, chosen so that several of them have strong similarities, while others have runs of repeated actions. Each action code generated a line segment of fixed length in the direction specified by the code. Only connected shapes were used, though disconnected shapes can be included easily by giving the simulated pen a degree of freedom perpendicular to the workspace. The model currently does not include any inertial terms so the direction changes instantaneously with each population code. The ID input was encoded by 50-bit binary vectors, with 5 randomly chosen bits set to 1 for each figure.

Figure 2 demonstrates the shapes that the system with $n = 10$ (i.e., $N_C = 300$, $N_R = 600$ and $N_I = 600$) draws for each of the training sequences after learning. Each figure was presented 20 times during learning, with random angular noise of $\pm 0.5^\circ$ for each line segment. Figure 3 shows performance when an angular error uniformly distributed in a range around 0° is added to each action segment during training. It demonstrates that the system is quite robust to such noise, mainly because of the use of WTA clusters in the sequence learner.

An important issue addressed in this paper is the system's ability to learn highly similar sequences, and the role of the selector system in doing so. This was studied by training the system on shapes with 20 segments each, such that the first 19 segments in each shape were almost identical (in a circular arrangement)

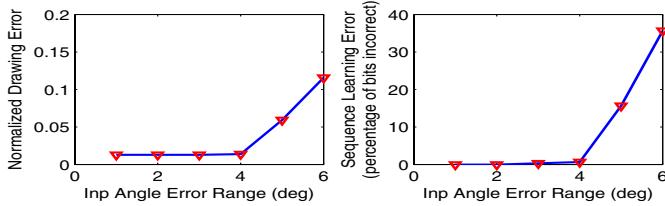


Fig. 3. Effect of angular input error in input on system performance: (a) Mean normalized angular drawing error/segment; (b) Percentage of incorrect bits in sequence recall

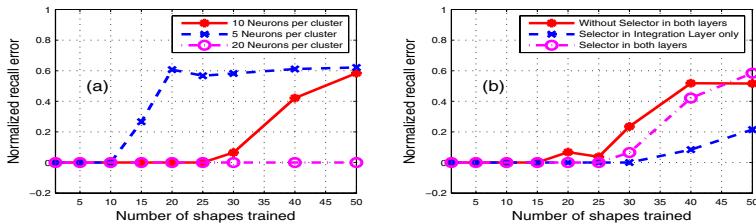


Fig. 4. Graph (a): Effect of varying cluster sizes on the capacity for learning near-identical shapes. Graph (b): Comparison of the system without cluster selection and those with cluster selection in Layer I and both layers I and R , while learning near-identical shapes.

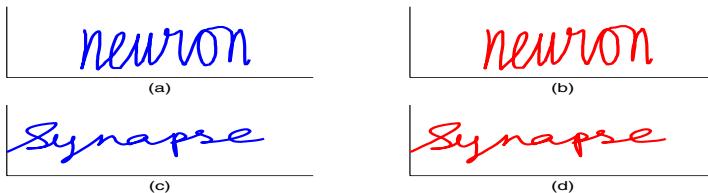


Fig. 5. Learning long and complex sequences in a network with $n = 20$. Graph (a, c): Training text; Graph (b, d): Recalled text.

and the only significant difference arose in the 20th segment. The shapes had distinct IDs. To learn these shapes, the system had to distinguish the internal representations of the sequences sufficiently. Figure 4(a) shows the results for storing up to 50 near-identical shapes in networks with $n = 5, 10$ and 20 . The results demonstrate that increasing the size of clusters increases the capacity for such learning, but even a small network can store several nearly identical sequences. Using the same training set (near-identical shapes), Figure 4(b) shows the comparative performance of a network without cluster selection, one with cluster selection only in layer I , and one with selection in both layers I and R . It is clear that ID-specific cluster selection provides significant increase in capacity

for learning similar sequences. Interestingly, the best performance is obtained when selection is applied only to layer I , presumably because disambiguation is important mainly for training the I -to- C connections.

Finally, Figure 5 shows the results when the same network is trained to store two very long and complex sequences representing two different words. Each sequence comprises several hundred steps (373 for “neuron” and 287 for “synapse”), and are stored simultaneously in the same network with $n = 20$. As shown in the figure, the sequences are recalled nearly perfectly after training.

5 Conclusion

In this paper, we have presented a model for learning population-coded kinematic sequences in a simple but biologically motivated neural system. In particular, our results show that, with the inclusion of a basal ganglia-like selector, the system is able to learn a large number of almost identical sequences, and many different types of shapes, including very complex ones. More detailed studies of the system, including the role of velocity coding, inertial factors, internal noise and downstream motor control will be presented in future papers.

Acknowledgement

This research was partly supported by a grant from the University of Cincinnati College of Engineering to the *Intelligent Autonomous Systems* research group.

References

1. Georgopoulos, A., Kalaska, J., Caminiti, R., Massey, J.: On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience* 2, 1527–1537 (1982)
2. Schwarz, A., Kettner, R., Georgopoulos, A.: Primate motor cortex and free arm movement to visual targets in 3-d space. i. relations between single cell discharge and direction of movement. *Journal of Neuroscience* 8, 2913–2927 (1988)
3. Mushiake, H., Inase, M., Tanji, J.: Neuronal activity in the primate premotor, supplementary and precentral motor cortex during visually guided and internally determined sequential movements. *Journal of Neurophysiology* 66, 705–718 (1991)
4. Schwartz, A.: Motor cortical activity during drawing movements: single unit activity during sinusoid tracing. *Journal of Neurophysiology* 68, 528–541 (1992)
5. Kakei, S., Hoffman, D., Strick, P.: Muscle and movement representations in the primary motor cortex. *Science* 285, 2136–2139 (1999)
6. Seitz, R., Stephan, K., Binkofski, F.: Control of action as mediated by the human frontal lobe. *Experimental Brain Research* 133, 71–80 (2000)
7. Shima, K., Tanji, J.: Neuronal activity in the supplementary and presupplementary motor areas for temporal organization of multiple movements. *Journal of Neurophysiology* 84, 2148–2160 (2000)
8. Kakei, S., Hoffman, D., Strick, P.: Direction of action is represented in the ventral premotor cortex. *Nature Neuroscience* 4, 1020–1025 (2001)

9. Nakamura, K., Sakai, K., Hikosaka, O.: Neuronal activity in medial frontal cortex during learning of sequential procedures. *Journal of Neurophysiology* 80, 2671–2687 (1998)
10. Averbeck, B., Chafee, M., Crowe, D., Georgopoulos, A.: Parallel processing of serial movements in prefrontal cortex. *Proceedings of the National Academy of Sciences USA* 99, 13172–13177 (2002)
11. Averbeck, B., Chafee, M., Crowe, D., Georgopoulos, A.: Neural activity in prefrontal cortex during copying geometrical shapes. i. single cells encode shape, sequence and metric parameters. *Experimental Brain Research* 150, 127–141 (2003)
12. Averbeck, B., Crowe, D., Chafee, M., Georgopoulos, A.: Neural activity in prefrontal cortex during copying geometrical shapes. ii. decoding shape segments from neuronal ensembles. *Experimental Brain Research* 150, 142–153 (2003)
13. d'Avella, A., Saltiel, P., Bizzi, E.: Combinations of muscle synergies in the construction of natural motor behavior. *Nature Neuroscience* 6, 300–308 (2003)
14. Lu, X., Ashe, J.: Anticipatory activity in primary motor cortex codes memorized movement sequences. *Neuron* 45, 967–973 (2005)
15. Graziano, M.: The organization of behavioral repertoire in motor cortex. *Annual Review of Neuroscience* 29, 105–134 (2006)
16. Ajemian, R., Bullock, D., Grossberg, S.: Kinematic coordinates in which motor cortical cells encode movement direction. *Journal of Neurophysiology* 84, 2191–2203 (2000)
17. Sergio, L., Kalaska, J.: Systematic changes in motor cortex cell activity with arm posture during directional isometric force generation. *Journal of Neurophysiology* 89, 212–228 (2003)
18. Cisek, P.: Cortical mechanisms of action selection: the affordance competition hypothesis. *Phil. Trans. R. Soc. B* 362, 1585–1599 (2007)
19. Matsuzaka, Y., Picard, N., Strick, P.: Skill representation in the primary motor cortex after long-term practice. *Journal of Neurophysiology* 97, 1819–1832 (2007)
20. Ajemian, R., Green, A., Bullock, D., Sergio, L., Kalaska, J., Grossberg, S.: Assessing the function of motor cortex: single-neuron models of how neural response is modulated by limb biomechanics. *Neuron* 58, 414–428 (2008)
21. Grillner, S.: Biological pattern generation: the cellular and computational logic of networks in motion. *Neuron* 52, 751–766 (2006)
22. Ijspeert, A., Crespi, A., Ryczko, D., Cabelguen, J.M.: From swimming to walking with a salamander robot driven by a spinal cord model. *Science* 315, 1416–1420 (2007)
23. Graybiel, A.: Building action repertoires: memory and learning functions of the basal ganglia. *Current Opinion in Neurobiology* 5, 733–741 (1995)
24. Lukashin, A., Georgopoulos, A.: A neural network for coding of trajectories by time series of neuronal population vectors. *Neural Computation* 6, 19–28 (1994)
25. Lukashin, A., Wilcox, G., Georgopoulos, A.: Overlapping neural networks for multiple motor engrams. *Proceedings of the National Academy of Sciences, USA* 91, 8651–8654 (1994)
26. Lukashin, A., Amirikian, B., Mozhaev, V., Wilcox, G., Georgopoulos, A.: Modeling motor cortical operations by an attractor network of stochastic neurons. *Biological Cybernetics* 74, 255–261 (1996)
27. Ans, B., Coitou, Y., Gilhodes, J.P., Velay, J.L.: A neural network model for temporal sequence learning and motor programming. *Neural Networks* 7, 1461–1476 (1994)
28. Bullock, D., Cisek, P., Grossberg, S.: Cortical networks for control of voluntary arm movements under variable force conditions. *Cerebral Cortex* 8, 48–62 (1998)

29. Taylor, J., Taylor, N.: Analysis of recurrent cortico-basal ganglia-thalamic loops for working memory. *Biological Cybernetics* 82, 415–432 (2000)
30. Taylor, N., Taylor, J.: Hard-wired models of working memory and temporal sequence storage and generation. *Neural Networks* 13, 201–224 (2000)
31. Grossberg, S., Paine, R.: A neural model of cortico-cerebellar interactions during attentive imitation and predictive learning of sequential handwriting movements. *Neural Networks* 13, 999–1046 (2000)
32. Doya, K.: What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks* 12, 961–974 (1999)
33. Houk, J.: Agents of the mind. *Biol. Cybern.* 92, 427–437 (2005)
34. Minai, A., Barrows, G., Levy, W.: Disambiguation of pattern sequences with recurrent networks. In: Proc. WCNN, San Diego, vol. IV, pp. 176–180 (1994)
35. Sun, R., Giles, C.: Sequence Learning: Paradigms, Algorithms, and Applications. Springer, Heidelberg (2001)
36. Reiss, M., Taylor, J.: Storing temporal sequences. *Neural Networks* 4, 773–787 (1991)
37. Doboli, S., Minai, A.: Using latent attractors to discern temporal order. In: Proceedings of IJCNN, Budapest, Hungary (July 2004)
38. Fujii, N., Graybiel, A.: Representation of action sequence boundaries by macaque prefrontal cortical neurons. *Science* 301, 1246–1749 (2003)
39. Jog, M., Kubota, Y., Connolly, C., Hillgaart, V., Graybiel, A.: Building neural representations of habits. *Science* 286, 1745–1749 (1999)
40. Matsuzaka, Y., Aizawa, H., Tanji, J.: A motor area rostral to the supplementary motor area (presupplementary motor area) in the monkey: neuronal activity during a learned motor task. *Journal of Neurophysiology* 68, 653–662 (1992)
41. Ghanem, A., Minai, A.: A modular gene regulatory network model of ontogenesis. In: Proc. Int. Conf. on Complex Sys., Boston, MA (2007)
42. Doboli, S., Minai, A., Brown, V.: Adaptive dynamic modularity in a connectionist model of context-dependent idea generation. In: Proceedings of the IJCNN 2007, Orlando, FL, pp. 2183–2188 (2007)
43. Minai, A., Iyer, L., Padur, D., Doboli, S.: A dynamic connectionist model of idea generation. In: Proceedings of the IJCNN 2009, Atlanta, GA (2009)
44. Perumal, S., Minai, A.: Stable-yet-switchable (sys) attractor networks. In: Proceedings of the IJCNN 2009, Atlanta, GA (2009)
45. Mountcastle, V.: The columnar organization of the neocortex. *Brain* 120, 701–722 (1997)
46. Widrow, B., Hoff, M.: Adaptive switching circuits. In: 1960 IRE WESCON Convention Record, Part 4, pp. 96–104 (1960)

Structural Analysis on STDP Neural Networks Using Complex Network Theory

Hideyuki Kato¹, Tohru Ikeguchi^{1,3}, and Kazuyuki Aihara^{2,3}

¹ Graduate School of Science and Engineering, Saitama University,
255 Shimo-Ohkubo, Sakura-ku, Saitama, 338-8570, Japan
kato@nls.ics.saitama-u.ac.jp,
tohru@ics.saitama-u.ac.jp

² Graduate School of Information Science and Technology, The University of Tokyo,
4-6-1 Komaba, Meguro-ku Tokyo, Japan
aihara@sat.t.u-tokyo.ac.jp

³ Aihara Complexity Modelling Project, ERATO, JST,
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan

Abstract. Synaptic plasticity is one of essential and central functions for the memory, the learning, and the development of the brains. Triggered by recent physiological experiments, the basic mechanisms of the spike-timing-dependent plasticity (STDP) have been widely analyzed in model studies. In this paper, we analyze complex structures in neural networks evolved by the STDP. In particular, we introduce the complex network theory to analyze spatiotemporal network structures constructed through the STDP. As a result, we show that nonrandom structures emerge in the neural network through the STDP.

1 Introduction

Billions of neurons exist in the brain and these neurons interconnect with each other. Using such complicated structures, many neurons act as an ensemble and realize effective information processing mechanisms in the neural networks. In such neural networks, it is generally believed that synaptic plasticity plays central roles for the memory, the learning, and the development of the brain. Then, the synaptic plasticity has been widely analyzed from both experimental and theoretical aspects.

During the last decade, the existence of the STDP has been clarified in several areas of the brains [1, 2, 3, 4, 5, 6]. Stimulated by these experimental results, the basic mechanisms of the STDP have been widely analyzed in model studies [7, 8, 9, 10, 11]. In particular, the synaptic distribution is experimentally and theoretically analyzed in detail. Results obtained from these analyses are important to understand the mechanisms of the STDP. However, it is also important to analyze spatiotemporal neural network structures. In this sense, we should analyze how the spatiotemporal structure would evolve in the STDP neural network.

From the viewpoint of the graph theory, until 1998, were mainly analyzed regular and random networks. The regular networks have a specific structure

or clusters. Then, distances between any two nodes are long or their radius becomes large. On the other hand, the random networks have small network radii and few clusters. However, it has been observed that real networks often have not only small network radii but also many clusters, which is called the small-world property [12]. Such a property ubiquitously exists in the real world, for example, the co-acting relationship in movie films, the power grid networks, and the anatomical structure of *C. elegans* [12]. On the other hand, the real networks often have scale-free structure whose degree distribution obeys the power law [13]. It is known that a scale-free property also universally exist in the real world. For example, the actor collaboration, the world wide web, and the power grid [13]. These network structures might be useful for the network activity or the information processing in the neural networks.

Then, in this paper, we analyze the spatiotemporal structure in the self-organized neural network through the STDP introducing the complex network theory.

2 Methods and Results

A neural network is constructed from the neuron model proposed by Izhikevich [14]. The model is given by 2-dimensional ordinary differential equations:

$$\dot{v} = 0.04v^2 + 5v + 140 - u + I, \quad (1)$$

$$\dot{u} = a(bv - u), \quad (2)$$

with the auxiliary after-spike resetting:

$$\text{if } v \geq 30 \text{ [mV]}, \text{ then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \quad (3)$$

where v represents the membrane potential of the neuron, u represents a membrane recovery variable, and $\cdot = d/dt$ where t represents time. The variable I represents the sum of random external inputs and synaptic currents. The neural network is heterogeneous, namely, it has excitatory and inhibitory neurons which have different dynamics in the neural network. In this paper, we use the regular spiking type neuron as 800 excitatory neurons while the fast spiking type neuron as 200 inhibitory ones. The parameters a , b , c , and d are set according to Ref. [14]. Each neuron connects to 100 postsynaptic neurons, however, no connections exist between any pairs of the inhibitory neurons. Excitatory synaptic connections have $1 \sim 10$ [ms] conduction delays which distribute uniformly in the neural network while all inhibitory synaptic connections have 1 [ms] conduction delay. Excitatory and inhibitory synaptic efficacy is set to 6 and -5 , respectively. In the neural network, the excitatory synaptic efficacy is modified through the STDP. We use a simple STDP function proposed by Song et al. [8] (Fig. 1(a)) and it is defined by

$$\Delta w = \begin{cases} A_p \cdot e^{-|\Delta t|/\tau_p} & (\text{if } \Delta t > 0) \\ -A_d \cdot e^{-|\Delta t|/\tau_d} & (\text{otherwise}) \end{cases} \quad (4)$$

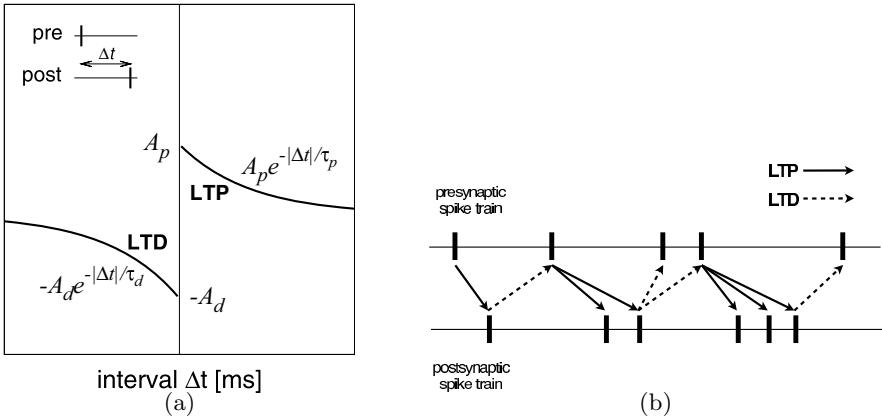


Fig. 1. (a) Functional form of spike-timing-dependent plasticity (STDP) with the parameters $A_p = 0.1$, $A_d = 0.12$, and $\tau_p = \tau_d = 20$ [ms]. (b) If a postsynaptic spike follows a presynaptic one (solid arrow), the synapse is potentiated (LTP). The reverse order (dashed arrow) results in depression (LTD).

where A_p and A_d are the learning rates of the long-term potentiation (LTP) and depression (LTD), and τ_p and τ_d are the time constants determining the exponential decays of the LTP and the LTD in the window function. In this equation, the variable Δt represents a relative spike timing between a pre- and a postsynaptic neurons. In the neural network, the nearest-neighbor implementation [15] is used, so that the nearest-neighbor spike pairs contribute for synaptic modifications (Fig. 1(b)). The STDP arises at every firing events and the synaptic efficacy is updated as $w \leftarrow w + \Delta w$ at every second. The synaptic efficacy is limited between 0 and 10: if, for the LTP, the efficacy becomes larger than 10, it is reset to 10. On the other hand, the efficacy is reset to 0 when the efficacy becomes smaller than 0 in the case of the LTD. This STDP neural network is driven by the Poissonian random inputs whose frequency is 10 [Hz]. In our simulation, the mean firing rate becomes about 17 [Hz] under the initial condition, however, the firing rate decreases from the initial condition and takes about 10 [Hz] at 100 [sec] (Fig. 2(a)). After that, the firing rate maintains about 10 [Hz]. Histograms of synaptic efficacy in both the initial condition and the equilibrium state are shown in Figs. 2(b) and (c), respectively.

Although all the synaptic efficacy takes the same value in the initial condition, synapses are competitive and their efficacy forms a bimodal distribution in the equilibrium state. The synaptic distribution is important for understanding the structure of the STDP neural network. However, it is not enough to reveal a hidden spatiotemporal structure. Then, to investigate the spatiotemporal structure of the STDP neural network in more detail, we introduce the complex network theory.

At first, we binarize the connectivity in the STDP neural network. Next, we generate an adjacency matrix $A = \{a_{ij}\}$ with a threshold value w_θ because we consider that strengthened synapses are important for the network activity.

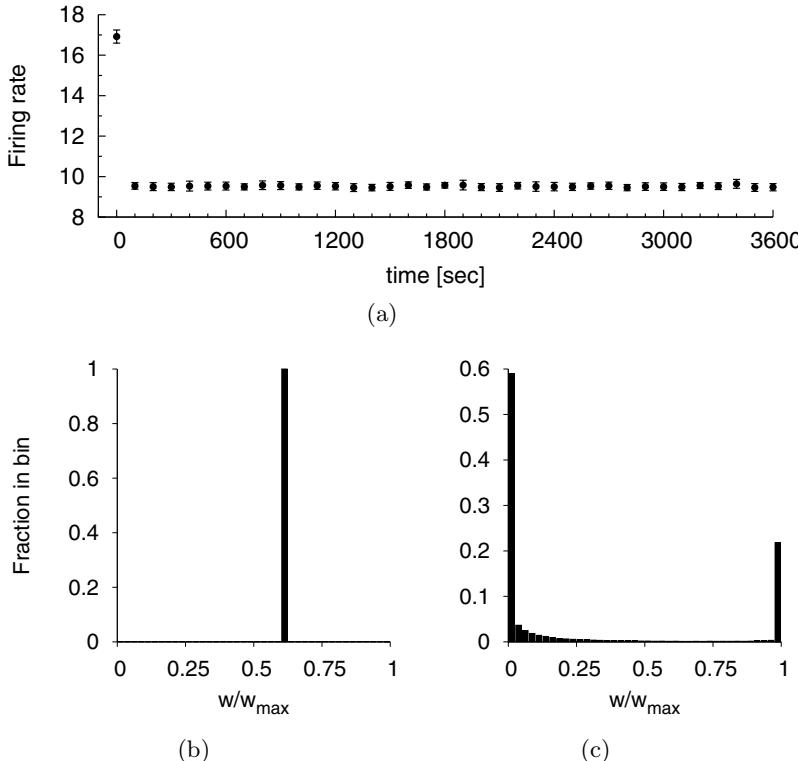


Fig. 2. (a) Temporal transition of the mean firing rate. The firing rates are plotted at every 100 seconds with the error-bars of the standard deviation for 20 simulations. Histograms of synaptic efficacy in (b) the initial condition and (c) the equilibrium state. The horizontal axis is divided into 40 bins.

Then, $a_{ij} = 1$ if the efficacy of synaptic connection from the i th and the j th neurons is no less than w_θ while $a_{ij} = 0$ if it is less than w_θ . In this paper, we set w_θ as 6, namely, it corresponds to the value of the initial excitatory synaptic efficacy. In addition, we attend the synapses only between excitatory neurons to simplify our analysis. For such a binarized neural network, we apply the complex network theory to analyze spatiotemporal structure of the STDP neural network.

Degree distribution

The degree distribution is one of the important statistics in the complex network theory. There are tree types of the degree: the indegree, the outdegree, and the mutual degree. The indegree and the outdegree of each neuron are defined as the number of afferent or efferent connections, respectively. The mutual degree is the sum of the indegree and the outdegree. If we describe the indegree and the outdegree of the i th neuron as k_i^{in} and k_i^{out} , its mutual degree is written as

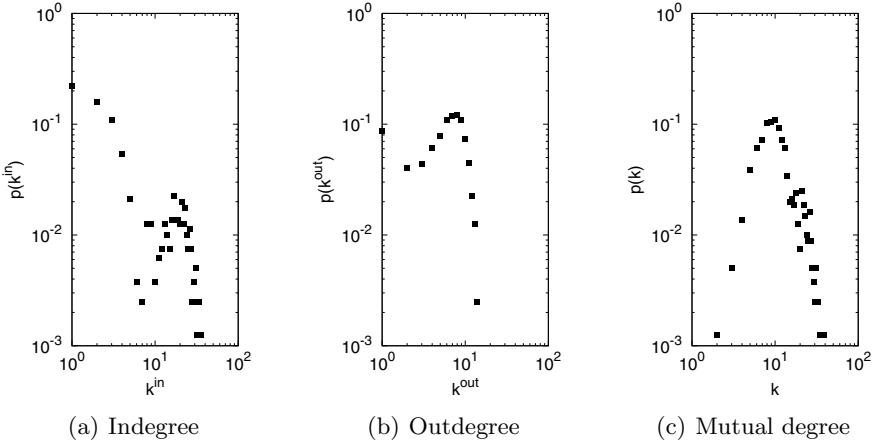


Fig. 3. Degree distribution of the STDP neural network in the equilibrium state in log-log scale

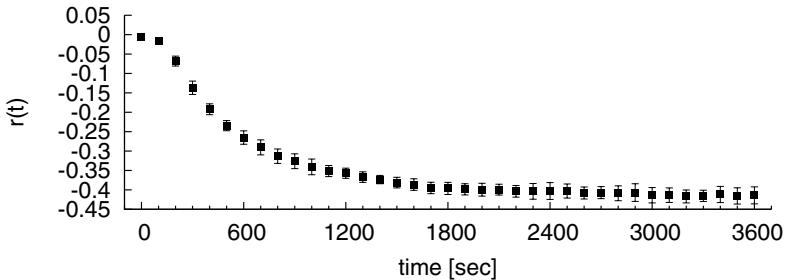


Fig. 4. Temporal transition of the assortativity coefficient in the STDP neural network. The figure is plotted at every 100 seconds.

$k_i = k_i^{in} + k_i^{out}$. Their distribution in the entire network may be inspected for the scale-free attributes such as power laws [13].

We can find that there exist two groups in the indegree distribution under or over $k^{in} = 10$ (Fig. 3(a)). The distribution of the high indegree obeys the Poissonian, which results from the network structure in the initial condition. In contrast, the distribution of the low indegree obeys the power law. These result indicate that the neural network organizes an incomplete scale-free structure because a complete scale-free structure is fragile if hub neurons are attacked. The STDP constructs such a structure to prevent the congestion of the information.

On the other hand, the outdegree distribution in the STDP neural network is similar to the Poissonian random. Although the indegrees of neurons distribute between 0 and 30 (Fig. 3(a)), the outdegree distribution is between 0 and 10

(Fig. 3(b)). This results mean that the neurons influence ten or so neurons in the neural network.

The mutual degrees in the STDP neural network also form a Poissonian random distribution. The range of their distribution is between 1 and 30 (Fig. 3(c)). From this result, the range of the mutual degree distribution is almost the same as that of the indegree one. Namely, this result indicates that neurons have low outdegrees but high indegrees. Thus, these neurons have small effect to the neurons. Then, the neural network controls its firing rate using these biased connections (Fig. 2(a)).

Assortative mixing

To evaluate the assortative mixing in the STDP neural network, we use the assortativity coefficient r [16] which is defined by

$$r = \frac{M^{-1} \sum_i j_i k_i - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2}{M^{-1} \sum_i \frac{1}{2}(j_i^2 + k_i^2) - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2} \quad (5)$$

where M is the number of connections in the network and j_i and k_i are the degrees of the neurons at the ends of the i th connection. This measure can take $-1 \leq r \leq 1$. It means that networks tend to be assortative if r is close to 1 while networks tend to be disassortative when r is close to -1 .

In the initial condition, the coefficient r takes about 0 because the network structure is almost random (Fig. 4). After that, the coefficient r gradually decreases and finally takes about -0.4 (Fig. 4). This result shows that the STDP leads disassortativity to the neural network. In addition, this tendency is the same as the neural network in Ref. [16]. However, comparing with disassortative networks in Ref. [16], the coefficient r of the STDP neural network is much lower. Then, the STDP induces much stronger disassortativity in the neural network.

Characteristic path length

The characteristic path length L is one of the important statistics as well as the degree in the complex network theory because this measure evaluates the small-world property: L is described as

$$L = \frac{1}{N(N-1)} \sum_{i,j} d_{ij} \quad (6)$$

where N is the number of neurons and d_{ij} is the shortest distance from the i th to j th neurons. In this case, the shortest distance corresponds to the minimum number of connections through which spikes pass from the i th neuron to the j th neuron.

To evaluate the small-world property, we should compare the characteristic path length of the STDP neural network with those of surrogate networks. In this paper, we use random networks as the surrogate networks. The surrogate networks are generated from the STDP neural network by randomly rewiring connections where both the indegree and the outdegree of the STDP neural

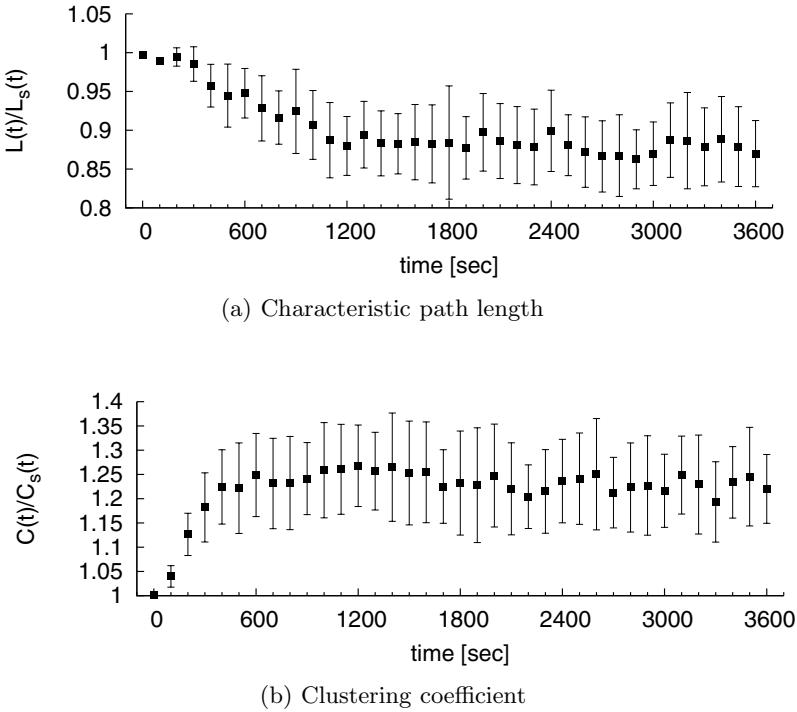


Fig. 5. Temporal transition of small-world property in the STDP neural network. Both the characteristic path length and the clustering coefficient of the STDP neural network are normalized by those of the surrogate network. We generate fifty surrogate networks. L_s and C_s are the mean values for the surrogate networks. Each figure is plotted at every 100 seconds.

network are preserved in the surrogate networks. In this paper, we generate fifty surrogate networks and use their mean value of the characteristic path length to evaluate that of the STDP neural network.

In the initial condition, $L(t)/L_s(t)$ takes about 1 (Fig. 5(a)). This result indicates that the neural network structure in the initial condition is random and it is similar to the result of the assortative mixing (Fig. 4). From the initial condition, $L(t)/L_s(t)$ decreases until 1,200 seconds and after that, $L(t)/L_s(t)$ converges to about 0.9. This result implies that the STDP neural network processes the information more effectively than the surrogate networks which have the same indegree and outdegree distributions.

Clustering coefficient

The clustering coefficient is also an important statistic to evaluate the small-world property in the complex network theory. The clustering coefficient is described by

$$C = \frac{1}{N} \sum_i C_i \quad (7)$$

where N is the number of neurons and C_i is the clustering coefficient of the i th neuron. The coefficient of the i th neuron is written as

$$C_i = \frac{m_i}{n_i(n_i - 1)}. \quad (8)$$

In Eq. (8), n_i is the number of adjacent neurons of the i th neuron and m_i is the number of connections between all the pairs of the adjacent neurons of the i th neuron. We calculate the clustering coefficient of the STDP neural network with normalization by the mean value of the clustering coefficient for the surrogate networks.

In the initial state, $C(t)/C_s(t)$ takes about 1 (Fig. 5(b)). This reason is the same as the results of both the assortative mixing and the characteristic path length (Figs. 4 and 5(a)). After that, $C(t)/C_s(t)$ gradually increases until 1,200 seconds and finally takes about 1.25. This result indicates that the STDP strengthens or weakens synapses to remain the clusters. Then, the STDP constructs neural network to be easy to share the information. Considering the result of the characteristic path length (Fig. 5(a)), it is clear that the STDP neural network has small-world property.

3 Conclusions

In this paper, we analyzed the spatiotemporal evolution of the structure in the self-organized neural network through the spike-timing-dependent plasticity (STDP) with the complex network theory. As a result, the STDP does not induce the scale-free property in the neural network. From the results of all the types of the degree, the outdegrees of high indegree neurons tend to become low. The outdegree of each neuron is no more than 20, which results in controlling the firing rate in the neural network. In other words, the neurons have low outdegrees if their indegrees become high, so that influence of their spikes is hardly to propagate for whole the neural network. These results corresponds to the result of the assortative mixing in the STDP neural network. In addition, it is also clarified that the STDP conducts the small-world structure in the neural network. This result indicates that the neural network shares the information and is possible to quickly process the information for whole the network. These results indicate that the neural network realizes not only the effective information processing but also robustness for attacking neurons. We believe that these unique structures are effective for the network activity and the information processing.

As a future work, we will analyze the relationship between these structures and the network dynamics in the STDP neural network to clarify how the spatiotemporal structures affects to the network activity and the information processing.

The research of T.I. is partially supported by Grant-in-Aid for Scientific Research (C) (No.20560352) from JSPS.

Bibliography

- [1] Markram, H., Lüboke, J., Frotscher, M., Sakmann, B.: Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs. *Science* 275, 213–215 (1997)
- [2] Bell, C.C., Han, V.Z., Sugawara, Y., Grant, K.: Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature* 387, 278–281 (1997)
- [3] Bi, G., Poo, M.: Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength and Postsynaptic Cell Type. *The Journal of Neuroscience* 18(24), 10464–10472 (1998)
- [4] Zhang, L.I., Tao, H.W., Holt, C.E., Harris, W.A., Ming Poo, M.: A Critical window for cooperation and competition among developing retinotectal synapses. *Nature* 395, 37–44 (1998)
- [5] Feldman, D.E.: Timing-Based LTP and LTD at Vertical Inputs to Layer II/III Pyramidal Cells in Rat Barrel Cortex. *Neuron* 27, 45–56 (2000)
- [6] Froemke, R.C., Dan, Y.: Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature* 416, 433–437 (2002)
- [7] Abbott, L.F., Nelson, S.B.: Synaptic plasticity: taming the beast. *Nature neuroscience supplement* 3, 1178–1183 (2000)
- [8] Song, S., Miller, K.D., Abbott, L.F.: Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience* 3(9), 919–926 (2000)
- [9] van Rossum, M.C.W., Bi, G.Q., Turrigiano, G.G.: Stable Hebbian Learning from Spike Timing-Dependent Plasticity. *The Journal of Neuroscience* 20(23), 8812–8821 (2000)
- [10] Rubin, J., Lee, D.D., Sompolinsky, H.: Equilibrium Properties of Temporally Asymmetric Hebbian Plasticity. *Physical Review Letters* 86(2), 364–367 (2001)
- [11] Gütig, R., Aharonov, R., Rotter, S., Sompolinsky, H.: Learning Input Correlations through Nonlinear Temporally Asymmetric Hebbian Plasticity. *The Journal of Neuroscience* 23(9), 3687–3714 (2003)
- [12] Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* 393, 440–442 (1998)
- [13] Barabási, A.L., Albert, R.: Emergence of Scaling in Random Networks. *Science* 286, 509–512 (1999)
- [14] Izhikevich, E.M.: Simple Model of Spiking Neurons. *IEEE Transactions on Neural Networks* 14(6), 1569–1572 (2003)
- [15] Izhikevich, E.M., Desai, N.S.: Relating STDP to BCM. *Neural Computation* 15, 1511–1523 (2003)
- [16] Newman, M.E.J.: Assortative mixing in networks. *Physical Review Letters* 89, 208701 (2002)

Time Coding of Input Strength Is Intrinsic to Synapses with Short Term Plasticity

Márton A. Hajnal

Eötvös Loránd University,
Pázmány Péter sétány 1/C,
1117, Budapest, Hungary
mahajnal@elte.hu
<http://nipg.inf.elte.hu/>

Abstract. Many neocortical synapses adapt their postsynaptic response to the input rate of the presynaptic neuron through different mechanisms of short term plasticity: Steady state postsynaptic firing rates become invariant to the presynaptic frequency. Still, timing may convey information about presynaptic rate: The postsynaptic current is shown here analytically to peak earlier when presynaptic input frequency increases. An approximate 1ms/10Hz coding sensitivity for AMPA, and 1ms/1Hz for NMDA receptors in post synaptic potentials was found by a multi-compartmental synapse simulation using detailed kinetic channel models. The slower the ion channels, the more expressed the time lag signal, but the same time the less the available headroom when compared at identical frequencies. Such timing code of input strength is transmitted most efficiently when postsynaptic amplitude is normalized by the input rate. Short term plasticity is a mechanism local to the synapse that provides such normalizing framework.

Keywords: Short term plasticity, frequency adaptation, timing, time code, time shift, time lag, neural code, synapse.

1 Introduction

Dynamical neural code, viewed as an intricate interplay of time and space patterns, far exceeds the information capacity of rate coding. Neurons extract information of and regulate each other by rate and time code patterns at different time and spatial scales, from ms responses to long term potentiation in time, and from synapses, through dendrite branches, to microcircuits and large interareal networks in space. For individual units of computation, e.g. synapses, dendrite branches and cell somata, integrative lowpass filters and coincident detector highpass filters correspond to utilize more the rate vs. the timing of their input pattern. Various adaptive normalization mechanisms are exhibited in order to tune the neurons' responses along the line between frequency and time domain, as in sensory gain control, information filtering, optimal cooperation, or maintainance of physiological operational range [1-3]. One form of adaptive normalization, short term synaptic plasticity, excercises its effect at the synapse

by presynaptic and postsynaptic mechanisms in the seconds time scale; that is between dynamic activity patterns and traditional long term plasticity [4–6].

For depressive synapses steady state postsynaptic firing rates become more or less invariant to the presynaptic frequency. It has recently been found that in the hippocampal CA3, cell subpopulations of pyramidal cells distributed in the same region are specialized for a given level of adaptation [7]. Certain cells have a frequency limit around 5–10Hz, below which they function as an integrating neuron, while above they fire independently of the input frequency [8, 9, 5]. However, even in this adaptive operational range, change in presynaptic firing rate seems to be detected by differential signaling of transients [5, 10].

In the present work only the steady states of rate-adapting short term depression are considered. They correspond to behaviour depending on average firing rate, not transients due to abrupt changes in firing rate. Such neurons in steady state will be shown in this paper to still possess rate information in their timings of post synaptic responses. A recent work examined rate information transfer through short term plasticity [10], though the model concentrated on filtering features. The present study examines a mechanism of rate information transfer that is intrinsic to synapses. It will be shown here that the timing of the peak of the excitatory post synaptic potential (EPSP) depends on the average frequency of the stimulus.

The conductance of ion channel receptors is determined both by the time-course of the neurotransmitter concentration in the synaptic cleft [11] and the probabilistic channel kinetics [12–15]. Macroscopic synaptic conductance can be approximated by fitting models to reproduce the ensemble average of the time course of single channel kinetics. At a phenomenological level of description, approximating conductance with two exponentials, the time lag found in this work primarily depends on the rise time of the receptor ion channel current; it will be shown here analytically. With more detailed modeling it turns out, that dendritic integration does accentuate a significant rate-dependent shift of the EPSP waveform. A conductance based multicompartmental model of a synapse with Markov kinetic channels was simulated to show time lag coding in realistic conditions. According to the channel kinetics, when leftover intrasynaptic glutamate, slower dissociation from receptor, and desensitization are taken into account, glutamate concentration change at different frequencies induces different timecourses of synaptic conductance.

2 Methods

2.1 Two-Exponential Approximation of Channel Current

For an analytical study of the time lag coding behaviour, the macroscopic conductance timecourse was approximated by a sum of two exponentials; usual with network simulations [16]:

$$P^*(t) = e^{-\frac{t}{\tau_d}} - e^{-\frac{t}{\tau_r}}, \quad (1)$$

where τ_r is the faster rising part of the waveform and τ_d decay time determines longterm behaviour. Macroscopic synaptic conductance is linearly related to the electrophysiological quantity of ensemble channel current, the excitatory post synaptic current (EPSC), when the membrane voltage is held constant, i.e. it is voltage-clamped to the resting potential.

For our purposes P^* is normalized to 1 not by the area under the curve, but by the peak value, P_{\max}^* . *Peak time* at $t = t_p$ is found by determining the maximum of the $P^*(t)$ curve by simple differentiation:

$$t_p = \frac{\tau_d \tau_r}{\tau_d - \tau_r} \log \frac{\tau_d}{\tau_r}, \quad (2)$$

$$P_{\max}^* = P^*(t_p) = \left(\frac{\tau_r}{\tau_d} \right)^{\frac{\tau_r}{\tau_d - \tau_r}} - \left(\frac{\tau_r}{\tau_d} \right)^{\frac{\tau_d}{\tau_d - \tau_r}}. \quad (3)$$

Note here, that any physiologically detailed mechanism of normalization by short term plasticity is represented in this normalization. The amplitude and waveform changes are separated and in the present framework only waveform changes considered determinants of timing code.

Previous inputs are represented by a periodic term, nT , with $T = 1/f$ input spike interval:

$$P_T^*(t) = \sum_{n=0}^{\infty} \left(e^{-\frac{t+nT}{\tau_d}} - e^{-\frac{t+nT}{\tau_r}} \right). \quad (4)$$

The input frequency dependence of the periodic peak time, $t_{T,p}$, will be demonstrated analytically in Section 3.1

2.2 Multicompartmental Synapse Model with Detailed Channel Kinetics

In order to incorporate realistic synaptic transmission effects, such as complex ion currents, membrane dynamics and receptor desensitization, a typical excitatory synaptic area with presynaptic bouton and postsynaptic dendritic spine was modeled. To drive the synaptic receptor ion channels, an injected current pulse (width 3ms) into a one-compartmental presynaptic axon terminal generated action potentials (APs) by Hodgkin-Huxley equations. Passive membrane resistance was $5k\Omega\text{cm}^2$, membrane capacitance was $1\mu\text{F}/\text{cm}^2$, resting potential was -70mV . The axon terminal was equipped with a voltage dependent two-state P/Q Ca^{2+} channel, a three state Ca^{2+} ATPase pump, and a four Ca^{2+} molecule-binding vesicle fusion factor. Subsequent glutamate concentration time-course in the synaptic cleft was modeled by a first order process. Model details and parameters as in [17].

A complex 21-state Markov kinetic AMPA receptor ion channel model was used to comply to recent measurements of AMPA EPSCs. Since the theoretical time lag effect was found to be very small for AMPARs (Section 3.1), minute details of physiological processes needed to be available. In this model the channel

is open in three states of 2, 3 and 4 molecules of bound glutamate. Bound states can desensitize both from open and closed channel states: five desensitized states for each four states of 1-4 bound glutamates. The average single channel conductance of 2, 3, and 4 molecule bound open states were 5, 7, and 12pS respectively. Model details and kinetic parameters as in [14].

A 5-state kinetic gating NMDA receptor model allowed to simulate NMDA EPSC time courses, with the free, 1 and 2 glutamate molecule binding states; 2-molecule glutamate bound state can change either to open or desensitized state. Mg²⁺ block was excluded from the simulations, represented as 0mM Mg²⁺ concentration, since the interest was focused on general EPSP waveforms. Model details by [18, 17] and parameters as in [19].

To study the effects of the AMPA and NMDA channel current, a pyramidal spine head was modeled based on measurements of [20–23], and the voltage was recorded. The diameter of the spine head was 0.5μm, membrane resistance was 12kΩcm², other passive parameters as for the presynaptic bouton. Voltage waveforms were approximately linear to currents; single synapse EPSP amplitudes were less than 10mV, with 0mV reversal potentials for channels.

The model was simulated in the NEURON environment with time resolution $dt = 0.005\text{ms}$. All kinetic models were implemented in equilibrium averages of probabilistic single-receptor states. Figures in this paper show one-periods of timecourses, taken when the simulation relaxed to a steady state with continuous periodic stimulation. For interfrequency comparison, waveforms were peak normalized, similar to (2) and (3).

3 Results

3.1 Analytical Solution for Input Frequency Transcoded to Time Lag

Since $\forall x > 0$, e^{-x} is less than unity, the infinite sums of the geometrical series for both terms in (4) are finite. With bases: $e^{-t/\tau_{d,r}}$, and multipliers at the n th iteration: $(e^{-T/\tau_{d,r}})^n$, the sum in (4) equals

$$P_T^*(t) = \frac{e^{-\frac{t}{\tau_d}}}{1 - e^{-\frac{T}{\tau_d}}} - \frac{e^{-\frac{t}{\tau_r}}}{1 - e^{-\frac{T}{\tau_r}}} . \quad (5)$$

The peak time at the periodic case, $t_{T,p}$, is found at the extremum of (5):

$$t_{T,p} = \frac{\tau_d \tau_r}{\tau_d - \tau_r} \left(\log \frac{\tau_d}{\tau_r} + \log \frac{1 - e^{-\frac{T}{\tau_d}}}{1 - e^{-\frac{T}{\tau_r}}} \right) . \quad (6)$$

The rightmost logarithmic term approaches zero (term inside log → 1) in the zero frequency ($T \rightarrow \infty$) limit, leading to (2). In the infinite frequency limit, i.e. for $T \ll \tau_r < \tau_d$, it approximates $\log \frac{\tau_r}{\tau_d}$:

$$\lim_{T \rightarrow 0} \frac{1 - e^{-\frac{T}{\tau_d}}}{1 - e^{-\frac{T}{\tau_r}}} = \frac{1 - 1 + \frac{T}{\tau_d}}{1 - 1 + \frac{T}{\tau_r}} = \frac{\tau_r}{\tau_d} ,$$

thus

$$t_{T,p}(T = 0) = \frac{\tau_d \tau_r}{\tau_d - \tau_r} \left(\log \frac{\tau_d}{\tau_r} + \log \frac{\tau_r}{\tau_d} \right) = 0; \quad (7)$$

the stimuli obviously become a constant current as the peak and the lowest point, $P_T(t = 0)$ join (Fig. 1a). To normalize the lowest point value, the peak value for the periodic case, $P_{T,\max}^*$, is needed. With $t = t_{T,p}$ from (6) substituted into (5):

$$P_{T,\max}^* = P_T^*(t_{T,p}) = \frac{e^{-\frac{t_{T,p}}{\tau_d}}}{1 - e^{-\frac{T}{\tau_d}}} - \frac{e^{-\frac{t_{T,p}}{\tau_r}}}{1 - e^{-\frac{T}{\tau_r}}}, \quad (8)$$

so the frequency dependence of the lowest conductance at the period boundaries, $t = 0$ or $t = T$, is

$$P_T(0) = \frac{1}{P_{T,\max}^*} \left(\frac{1}{1 - e^{-\frac{T}{\tau_d}}} - \frac{1}{1 - e^{-\frac{T}{\tau_r}}} \right). \quad (9)$$

In the zero frequency ($T \rightarrow \infty$) limit (9) reduces to 0 (with (8) \rightarrow const.). Thus, at low frequencies, any incoming spike would meet low baseline activity: the headroom is large, allowing for optimal signal to noise ratio. In the infinite frequency limit $P_T(0) \rightarrow 1$ as $T \rightarrow 0$. It means that at higher frequencies when the headroom shrinks to the range of noise, the possibility to propagate timing information saturates (Fig. 1b).

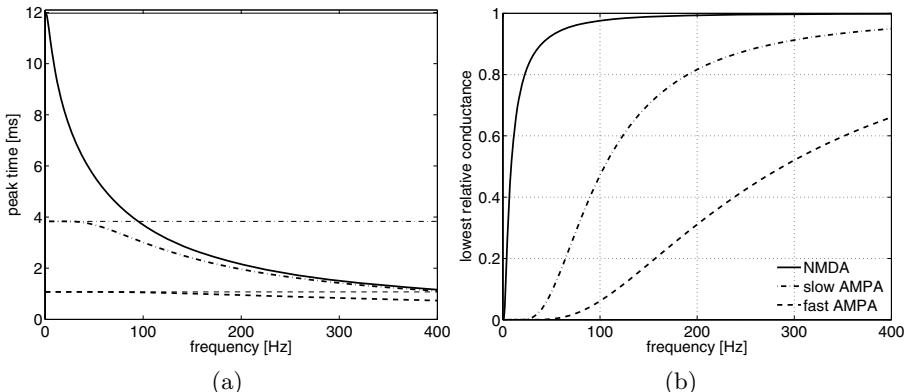


Fig. 1. Timings of peak conductance, $t_{T,p}$ (a), and the lowest values of conductance (b) depend on stimulus frequency. Analytical solutions for conductance approximated with two exponentials. NMDA (solid, $\tau_r=5\text{ms}$, $\tau_d=150\text{ms}$), slow AMPA (dash-dot, $\tau_r=3\text{ms}$, $\tau_d=5\text{ms}$), fast AMPA (dashed, $\tau_r=0.5\text{ms}$, $\tau_d=5\text{ms}$). Thin lines in (a) show the corresponding zero frequency limit peak time. Conductance saturation points in (b) depend on rise and decay time.

3.2 Simulation

The conjecture that rates appear in timing, suggested by the analytical solution, was confirmed by simulating the detailed models. Typical waveform shifts to frequency change are depicted in Fig. 2a-d; the long synaptic delays are visible, as the reference frames were set by the onsets of periodic presynaptic stimuli. Both EPSCs and EPSPs were subsequently shifted to earlier onsets from above 5-10Hz, while below, both positive and negative correlation, i.e. hysteresis, was found (Fig. 3a-b). Compared to the theoretical solutions of (6) the simulations show significantly larger time shifts to frequency change; even for the fast AMPA receptors, a range of 2.5ms can be observed within 25Hz, or five fold, input rate change between 3-27Hz. By linear approximation the sensitivity is $\approx 1\text{ms}/10\text{Hz}$ for AMPA, and $\approx 1\text{ms}/1\text{Hz}$ for NMDA receptor induced EPSPs. The saturation of available headroom, (9) and Fig. 1b, was also expressed by the simulation (Fig. 2b,d).

4 Discussion

In this paper it has been shown that the presynaptic rate could and might be transferred to postsynaptic time lag. Though direct experiments would help evidencing the hypothesis, biological indications do abound. The time scale of sensitivity to differences in timings and phases is supposedly very small: Precision requirements for auditory path processing [24], timing of synaptic multiquantal release [25], phases of a multi unit activity formation [26] are all within the range of a few milliseconds. These data might raise the relevance of the less expressed AMPA time lag as well. Considering jitter, noise in timings of spikes, a small increase in precision could nonlinearly amplify signal to noise ratios [25].

The details of dendritic computation and collective neuronal behaviour can alter the efficiency of time lag coding. When dendrites require inputs with more synchrony and exhibit dendritic spikes [27], the propagation of timing information to more proximal dendrite branch regions is enhanced. Synchronization also raises the possibility to more effectively exploit the smaller AMPA time lag changes. Neural code emerges when the receiver neuron can utilize the information transmitted. Time lag code may emerge if timing of input spikes onto a dendrite branch is effectively used for coincidence summation. However, not all modes of neuronal operations allow for coding in this time lag. Noisy, high average current and dendritic equalization [28], or induced bursts [29] make such timing information lost to more proximal dendrite regions. Rate sensitivity for low frequencies and timing sensitivity for higher frequencies might be alternative modes of neuronal information processing.

When the inputs are more periodic, timing has a fixed framework; phase code emerges. Phase coding is supposedly found more often when synchrony is stabilized by local inhibitory (i.e. basket cells, multi unit activity) or interareal oscillator-seed (i.e. septal nuclei, entorhinal cortex) connections, corresponding to functioning in the context of gamma and theta cycles [30]. In such context, input rate is transcoded to the phase within a theta or gamma cycle, by the

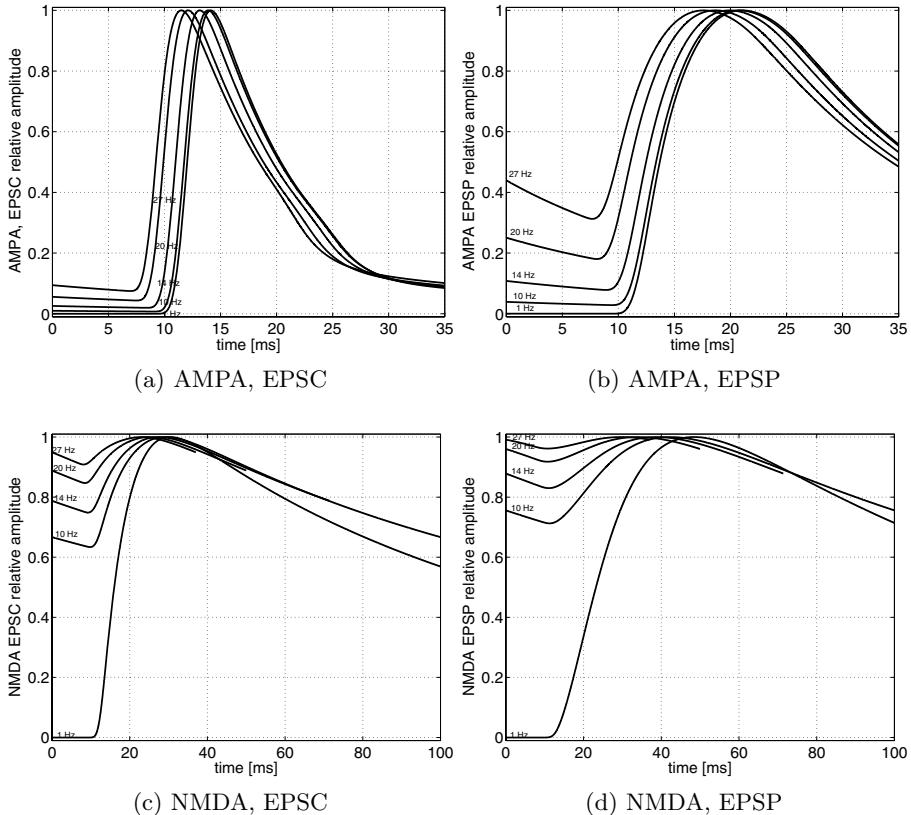


Fig. 2. Waveforms of AMPA (a,b) and NMDA (c,d) receptor channel EPSCs (a,c) and spine head EPSPs (b,d) at 1, 10, 14, 20 and 27 Hz stimuli. Multicompartmental simulations of kinetic channel models. The timeframe is fixed to presynaptic stimulation onsets at $t = 0$ and $t = T$; shorter curves: one period shown.

rhythmic rise and fall of baseline activity [31, 26]. The importance of the results in the present paper, though, comes from the observation, that, without any external cyclic change in baseline level, the time lag code is located in one-synapse mechanisms.

Furthermore, the intrinsic timing code of input strength works in consort with short term plasticity. Normalization would prevent the functioning of phase coding of input strength in a theta or gamma background, while normalization would enhance it in the presented intrinsic time lag-shift code. The same mechanism is also the limitation of the time lag code: When normalization is not sufficient, current driven action potential generations hide timing information. Nevertheless, short term depression is suspected to become inversely proportional to higher stimulus frequency [5, 9], intrinsically leading to correct normalization.

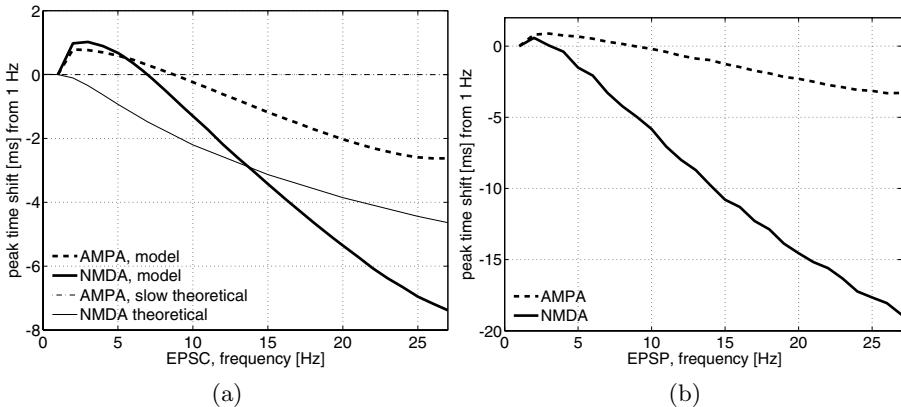


Fig. 3. Stimulus frequency dependence of peak timing shifts, referenced to the peak time at 1Hz. Negative values mean earlier peaking. Simulations of the multicompartmental, kinetic channel models, relative EPSCs (a) and EPSPs (b). EPSCs are compared to analytical solutions (thin lines) from (6) as shown in Fig. 1b. Simulations exhibit hysteresis in the lower, <5-10Hz, frequency range.

So far only dynamic neuronal patterns have been considered. Time-lag coded rate can also be of importance in learning and long term plasticity. It seems that coding and plasticity are inherently interrelated. Evidence is gathering, that short term plasticity may retain recent information about presynaptic spikes in Ca^{2+} accumulation [32]. Since the normalizing effect of short term plasticity is a requirement of effective post synaptic time lag code, postsynaptic timing may contain information about presynaptic Ca^{2+} levels as well. In addition, spike time dependent plasticity [33] is directly enhanced via these findings. At higher frequencies the shortened postsynaptic onset of dendritic and somatic APs may consequently result in quicker arrivals of backpropagating APs. Note that the time lag code effect is much more salient with the slower NMDA receptors that are activated mainly when backpropagating APs are present.

To conclude, the postsynaptic peak time consists of the sum of two time delays. One is a relatively fixed synaptic delay from the timing of the presynaptic spike, depending on synapse and membrane conductance properties. Second, as in this paper, it might code the average frequency of the presynaptic spiketrain in the scale of milliseconds changes in this delay. It may seem that individual neurons are tuned to specific information coding strategy between frequency domain – time domain preference by the extent of short term plasticity.

Acknowledgements. The author has been supported by the PhD program at Eötvös Loránd University, the ENNS Travel Grant and the National Office for Research and Technology (NKTH), Hungary.

Bibliography

- [1] Chance, F.S., Abbott, L.F., Reyes, A.D.: Gain modulation from background synaptic input. *Neuron* 35(4), 773–782 (2002)
- [2] Abbott, L.F., Regehr, W.G.: Synaptic computation. *Nature* 431(7010), 796–803 (2004)
- [3] Wark, B., Lundstrom, B.N., Fairhall, A.: Sensory adaptation. *Current Opinion in Neurobiology* 17(4), 423–429 (2007)
- [4] Zucker, R.S., Regehr, W.G.: Short-term synaptic plasticity. *Annual Review of Physiology* 64(1), 355–405 (2002)
- [5] Grande, L.A., Spain, W.J.: Synaptic depression as a timing device. *Physiology* 20(3), 201–210 (2005)
- [6] Yang, H., Xu-Friedman, M.A.: Relative roles of different mechanisms of depression at the mouse endbulb of held. *Journal of Neurophysiology* 99(5), 2510–2521 (2008)
- [7] Hemond, P., Epstein, D., Boley, A., Migliore, M., Ascoli, G.A., Jaffe, D.B.: Distinct classes of pyramidal cells exhibit mutually exclusive firing patterns in hippocampal area CA3b. *Hippocampus* 18(4), 411–424 (2008)
- [8] Varela, J.A., Sen, K., Gibson, J., Fost, J., Abbott, L.F., Nelson, S.B.: A quantitative description of short-term plasticity at excitatory synapses in layer 2/3 of rat primary visual cortex. *Journal of Neuroscience* 17(20), 7926–7940 (1997)
- [9] Fuhrmann, G., Segev, I., Markram, H., Tsodyks, M.: Coding of temporal information by activity-dependent synapses. *Journal of Neurophysiology* 87(1), 140–148 (2002)
- [10] Graham, B., Stricker, C.: Short term plasticity provides temporal filtering at chemical synapses. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 268–276. Springer, Heidelberg (2008)
- [11] Clements, J., Lester, R., Tong, G., Jahr, C., Westbrook, G.: The time course of glutamate in the synaptic cleft. *Science* 258(5087), 1498–1501 (1992)
- [12] Partin, K.M., Fleck, M.W., Mayer, M.L.: AMPA receptor flip/flop mutants affecting deactivation, desensitization, and modulation by cyclothiazide, aniracetam, and thiocyanate. *Journal of Neuroscience* 16(21), 6634–6647 (1996)
- [13] Dingledine, R., Borges, K., Bowie, D., Traynelis, S.F.: The glutamate receptor ion channels. *Pharmacological Reviews* 51(1), 7–62 (1999)
- [14] Raghavachari, S., Lisman, J.E.: Properties of quantal transmission at cal synapses. *Journal of Neurophysiology* 92(4), 2456–2467 (2004)
- [15] Zhang, W., Howe, J.R., Popescu, G.K.: Distinct gating modes determine the biphasic relaxation of NMDA receptor currents. *Nature Neuroscience* 11(12), 1373–1375 (2008)
- [16] Dayan, P., Abbott, L.F.: Theoretical Neuroscience. MIT Press, Cambridge (2000)
- [17] Destexhe, A., Mainen, Z., Sejnowski, T.: Kinetic models of synaptic transmission. In: Methods in Neuronal Modeling, pp. 1–25. MIT press, Cambridge (1998)
- [18] Clements, J.D., Westbrook, G.L.: Activation kinetics reveal the number of glutamate and glycine binding sites on the N-methyl-aspartate receptor. *Neuron* 7(4), 605–613 (1991)
- [19] Destexhe, A., Mainen, Z.F., Senowski, T.J.: An efficient method for computing synaptic conductances based on a kinetic-model of receptor-binding. *Neural Computation* 6(1), 14–18 (1994)
- [20] Segev, I., Rall, W.: Computational study of an excitable dendritic spine. *Journal of Neurophysiology* 60(2), 499–523 (1988)

- [21] Matsuzaki, M., Ellis-Davies, G.C.R., Nemoto, T., Miyashita, Y., Iino, M., Kasai, H.: Dendritic spine geometry is critical for ampa receptor expression in hippocampal ca1 pyramidal neurons. *Nature Neuroscience* 4(11), 1086–1092 (2001)
- [22] Tsay, D., Yuste, R.: On the electrical function of dendritic spines. *Trends in Neurosciences* 27(2), 77–83 (2004)
- [23] Araya, R., Eisenthal, K.B., Yuste, R.: Dendritic spines linearize the summation of excitatory potentials. *Proceedings of the National Academy of Sciences* 103(49), 18799–18804 (2006)
- [24] Elhilali, M., Fritz, J.B., Klein, D.J., Simon, J.Z., Shamma, S.A.: Dynamics of precise spike timing in primary auditory cortex. *Journal of Neuroscience* 24(5), 1159–1172 (2004)
- [25] Gasparini, S., Migliore, M., Magee, J.C.: On the initiation and propagation of dendritic spikes in CA1 pyramidal neurons. *Journal of Neuroscience* 24(49), 11046–11056 (2004)
- [26] Womelsdorf, T., Schöffelen, J.M., Oostenveld, R., Singer, W., Desimone, R., Engel, A.K., Fries, P.: Modulation of neuronal interactions through neuronal synchronization. *Science* 316(5831), 1609–1612 (2007)
- [27] Spruston, N.: Pyramidal neurons: dendritic structure and synaptic integration. *Nature Reviews Neuroscience* 9(3), 206–221 (2008)
- [28] Rumsey, C.C., Abbott, L.F.: Synaptic democracy in active dendrites. *Journal of Neurophysiology* 96(5), 2307–2318 (2006)
- [29] Chadderton, P., Margrie, T.W., Häusser, M.: Integration of quanta in cerebellar granule cells during sensory processing. *Nature* 428(6985), 856–860 (2004)
- [30] Womelsdorf, T., Fries, P.: The role of neuronal synchronization in selective attention. *Current Opinion in Neurobiology* 17(2), 154–160 (2007)
- [31] Fries, P., Nikolic, D., Singer, W.: The gamma cycle. *Trends in Neurosciences* 30(7), 309–316 (2007)
- [32] Mongillo, G., Barak, O., Tsodyks, M.: Synaptic theory of working memory. *Science* 319(5869), 1543–1546 (2008)
- [33] Song, S., Miller, K.D., Abbott, L.F.: Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience* 3(9), 919–926 (2000)

Information Processing and Timing Mechanisms in Vision

Andrea Guazzini¹, Pietro Lió², Andrea Passarella¹, and Marco Conti¹

¹ Institute of Informatics and Telematics - IIT - CNR,
via G. Moruzzi, 1 - 56124 Pisa, Italy

{andrea.guazzini, andrea.passarella, marco.conti}@iit.cnr.it

² Computer Laboratory, University of Cambridge, 15, J.J. Thompson Avenue,
Cambridge, CB30FD, UK
pl219@cam.ac.uk

Abstract. Researches of neural mechanism of time perception is one of the fastest growing areas of neuroscience. The visual system presents several examples of timing mechanisms. Its activity is characterized by a complex network of synchronized elements which cooperate together. Some authors recently proposed that neural circuits should be inherently capable of temporal processing as a result of the natural complexity of cortical networks coupled with the presence of time-dependent network properties. We present an adaptive feedback model which, through a temporal-to-spatial transformation is able to explain recent experiments on the relationships between vision and time/space perception.

1 Introduction

A fundamental question about the perception of time is whether the neural mechanisms underlying temporal judgments are universal and centralized in the brain or modality specific and distributed. Some authors have recently hypothesized that the temporal scaling of visual experience and the temporal tuning properties of certain visual neurons may be linked [3]. The apparent duration of a dynamic stimulus can be manipulated in a local region of visual space by adapting to oscillatory motion or flicker [10]. Perceived time is compressed when stimuli are flashed shortly before or after the onset of a saccadic eye movement [5]. Neurons in visual areas of primate parietal cortex have reduced latencies to visual stimulation at the time of a saccade [9]. Recently one class of models, called state-dependent networks, has been developed in order to demonstrate that a neural circuits can be inherently capable of temporal processing as a result of the natural complexity of cortical networks coupled with the presence of time-dependent network properties [11]. We present an adaptive feedback model which we show to provide a possible explanation of recent experiments and insightful clues in order to elucidate the relationships between stimulus neural encoding and time/space perception [13]. Our model can be defined as an adaptive feedback model because it changes the strength of network connections during its activity in order to adapt to the changing contexts during the process; modifying retroactively its sensitivity to the subsequent stimuli.

Brain timing and time Perception

From a neuropsychological point of view, interval timing in the range of milliseconds to minutes is affected in a variety of neurological populations involving disruption of the frontal cortex, hippocampus, basal ganglia and cerebellum. The conclusion is that representation of time depends on the integration of multiple neural systems [1]. Models of time perception all agree that temporal information is processed in many ways: it is remembered, compared to other temporal information, and used in the production of motor outputs. Researchers have had great difficulty in pinning timing-related activity in the brain to any specific type of function. This is largely because most of the time measurement tasks draw upon more than one process, making it difficult to tease the various components apart [2]. Johnston et al. has shown that apparent duration of a dynamic stimulus can be manipulated in a local region of visual space by adapting to oscillatory motion or flicker [10]. This implies spatially localized temporal mechanisms in duration perception. The authors have not found concomitant changes in time of onset or offset of the test pattern, demonstrating a direct local effect on the time duration perception rather than an indirect effect on the time course of neural processing. Moreover many perceptive tasks do not require explicit encoding of time: perception of visual motion, for example, relies on the output of spatio-temporally tuned neurons rather than on independent estimates of space and time. While there is good evidence about different clocks for different interval lengths, some recent evidences point clearly to the existence of visually based timing mechanism [6]. Finally one of the most up to date experimental result concerns the existence of a large contrast effects in the discrimination of short temporal intervals. Della Rocca et al. examined the effect of temporal distractors on interval discrimination. Subjects compared the duration of the second interval in a three-flash sequence with the interval between a two-flash sequence. For short durations, the presence of the irrelevant distractor interval affected the apparent duration of the test duration in a contrasting manner: short distractors caused it to appear longer, and vice versa. For very short probe durations ($< 100ms$) the effect was large, changing the perceived duration by up to a factor of two. For longer probe durations the effect of distractors reduced steadily, to no effect for durations greater than 500 ms . The results for visual flashes, auditory tones, and brief vibrations were similar, implying the existence of two mechanisms for timing brief events, one for short intervals (less than 500 ms), and another for longer intervals, and that the transition between the two is gradual [15]. One hypothesis drawn from the previous evidences is that cerebral circuits are inherently able to rescale durations in a proportional manner and compensate for the error differences generated by the cerebellum. With the motivation of modeling some of these experimental findings, we organize the paper as follows. In the next section we provide a description of the reasoning and mathematical formulation of the model. In section results we discuss the extensive simulations we have performed. Then the following section reports on conclusions. The software used for the model is available upon request from the first author.

2 The Model

We have modeled a key mechanism for the encoding of the temporal characteristics of the neural visual stimuli. Our inspiration originates from works on neural and visual timing. Several hypothesis of time measuring systems, peculiar to neural circuits, demonstrates integration (synchronization) enables both the evaluation of different time scales and the binding between different sensorial/cognitive modalities. There is a general belief on the hypothesis of the existence of a representation of temporal information relative to visual stimuli. The fundamental elements of the representation of visual information should be able to encode temporal information similarly to the other characteristics of the percept (color, frequency, brightness). In a recent paper, Buonomano proposed that short-term plasticity and dynamic changes in the balance of excitatory-inhibitory interactions may underlie the decoding of temporal information, that is, the generation of temporally selective neurons [3]. He first showed that it is possible to tune cells to respond selectively to different intervals by changing the synaptic weights of different synapses in parallel. Short-term plasticity is a usage-dependent change in synaptic strength on the time scale of millisecond to seconds and is observed in almost every synapse types of the central nervous system. Each type of synapse has its own specificity with respect to this property. When stimulated a few times within a second, some synapses show facilitation, others depression or else complex sequences of facilitatory and depressing changes [7].

Behavior of a chain of neurons: the neural delay chain

A neuron that responds to a vertical bar or a 1 kHz tone must receive functional inputs from the appropriate sensory neurons in the retina or cochlea, while temporal information refers to stimuli defined by the temporal structure of active sensory neurons. If a bar of light is present for 50 or 100 msec, in both cases the same groups of retinal ganglion cells are active. Similarly, if two brief 1 kHz tones are separated by 50 or 100 msec, the same population of hair cells will be active. Thus, for neurons to respond selectively to a 50 or 100 msec stimulus, an additional process such as a temporal to spatial transformation must occur. Several experimental evidences suggest that the neural signal encode "time related properties" by changing the state of the network's connections. Thus the key mechanism could rely with the propagation of the signal in the earlier states of the neural encoding process ($\lesssim 1 \text{ sec}$). This networks could be regarded as "neural delay chain", and be represented by a neural network where the signals propagates from the primary perceptive circuits. It is possible to identify a system characterized by computationally well-definable elements which represent the neural delay chain. In such system the microscopic elements may be characterized following the formalism of the neural networks typified by synapses with long -term plasticity, which tend to synchronize their activity. Consequently the activity may be treated at numerical level as continuous value instead of considering the discharge frequency of the single nodes. Therefore each node may

be characterized by an activation value between 0 and 1 which represents the synchronization level of the node's neurons. Each node is connected through oriented synapses to the following node. Due to the short term plasticity property of synapses it is possible to represent their level of strengthening and their status after the passage of the neural spike. Figure 1 shows the effects of a propagation of a neural signal along a neural delay chain. In the upper part of the figure the ordinate represents the activity of the node, while in lower part the ordinate indicates each synaptic weight along the chain. The abscissa represents the neural chain which carry out the role of a temporal buffer where the propagation of the neural signal encodes time information about the stimulus. The elements (x_i), defined simply "nodes", represent thus populations of neurons, and the activation value associated to each node ($\in [0, 1]$) represents the synchronization status of the node's neurons network. This approach allows to study a neural delay chain using a continuous representation of the neural signal. The status of the connections of the node is explicitly taken into account the status of the connections of the nodes (a_i). The neural spike generated by the visual stimulus propagates along the chain, encoding the temporal information about the stimulus. The activation state of this network could be read by output neurons in order to assess the duration of a stimulus or of an interval, as it happens for distances between objects and their size. In this way we obtain in our model a temporal to spatial transformation.

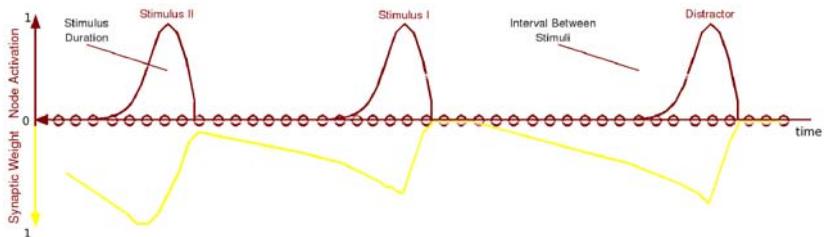


Fig. 1. The neural spike of the visual stimulus propagates along the neural delay chain (from left to the right) and encodes implicitly the time information about the stimulus. The activation state of this neural network is read out by specific neurons which assess the duration of a stimulus or of an interval, the same as it happens for distances among objects and their size.

We model a neural network where each node (x) is characterized by an activity value (i.e. the state of synchronization) between 0 and 1. Each node has excitatory synapses (a) toward the following one. Synapses evolve their weight dynamically according to the connected nodes' activity. The coupled dynamical equations which describe the system evolution are characterized by three control parameters: A synaptic weight decay factor (α), a learning factor (γ) which rules the synapses weight increment due to its activity, and a parameter which links the wave propagation velocity with the synaptic weight (θ). Each node within

the delay chain must possess an activation function temporally correlated to the one of the preceding node. In this sense its activation must be proportional to the activation of the previous node properly weighted by the connection (a_{ij}). Furthermore this equation determines the speed of the neural impulse as a function of θ , as shown by the Figure 2(b) where the normalized wave velocity is indicated in the ordinate as a function of a fixed synaptic weights characterizing the neural delay chain. Specifically the value of activation of the node at the position $i+1$ at the time $t+1$ can be written as follows:

$$x_{i+1}^{t+1} = x_i^t + (a_i^t)^\theta (x_i^t - K_x). \quad (1)$$

The synapse must enforce itself very rapidly depending on the signal coming from the preceding node and directed to the following one. The learning function of these oriented synapses must therefore be very sensitive to the activation changes of the following node and must be able as well to strengthen itself fast enough so as to facilitate the propagation of the activation wave. In order to approximate the non-linear features of the synaptic dynamics we use an hyperbolic tangent (1), whose weight grows as a function of the learning factor (γ) and the signal intensity. Finally the parameter ε is here introduced to seed the dynamics and so driving its initial regime. The value of synapse i at the time $t+1$ can be computed as follows:

$$a_i^{t+1} = a_i^t + \tanh\left(\frac{(a_i^t - K_a)}{(1 - K_a)} + \varepsilon\right) \cdot \gamma \cdot (x_i^t - K_x). \quad (2)$$

In Figure 2(a) is reported the value of a synaptic weight after the passage of neural signals with respect to the same value before the signal, for different stimulus duration. The return of the synapse to the rest condition (i.e. when no signals are exciting the synapse) should be sufficiently fast on its sensitiveness and in order to increase the precision of the system in the resolution of different signals. This dynamics can be modeled as follows:

$$a_i^{t+1} = a_i^t + \alpha \cdot (a_i^t - K_a). \quad (3)$$

Since we are dealing with a symbolic representation of a biological system it is appropriate to define minimum activity thresholds for the elements of the system. So we will have a basic threshold of activity for the nodes (K_x) and a rest value for the synapses (K_a).

Our model relies on the growing and decay dynamics of the synaptic weights, and the interplay between these aspects and the propagation/diffusion of the neural signals. Consequently for simplicity we have explicitly separate the decay dynamics from the growing one.

$$\begin{cases} \text{if } (x_i^t - K_x) > 0 \rightarrow \text{equation 2 holds,} \\ \text{if } (x_i^t - K_x) = 0 \rightarrow \text{equation 3 holds,} \end{cases}$$

The first equation determines the synapses behaviour when no signal is arriving, in this case the decay rate is proportional both to the decay factor α and to the value of the synaptic weight itself. The second equation model the dynamics of synapses where a signal is arriving.

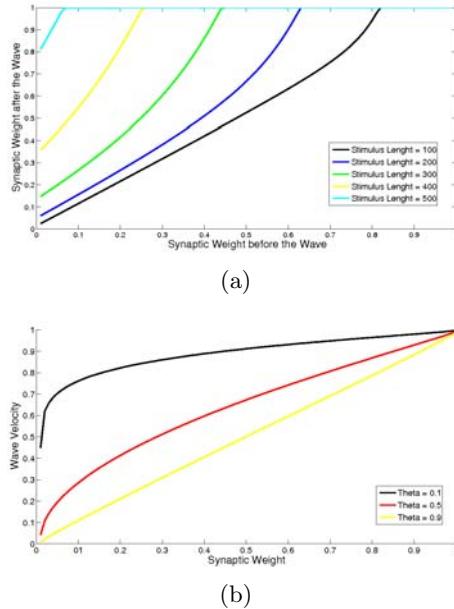


Fig. 2. (a)-Synaptic weight increase for different stimulus lengths; (b)-Normalized wave velocity for different values of the parameter which links the wave propagation velocity with the synaptic weight (θ)

Numerical simulation

From the experimental data about the effects of a distractor on the perception of the time intervals, the existence of three distinct regimes is pointed out [15]. One where the distractor is too far to interfere with the couple of stimuli; a second where the distractor is very close to the first stimulus of the couple, thus causing an apparent dilatation of the perceived interval between the subsequent stimuli. Finally, a third regime where the distance of the distractor from the first stimulus is intermediate between the other two cases, and where a contraction of the perceived interval is observed. Our interpretation is that this phenomenon is generated by the nonlinearities which characterize the functions of the wave propagation, and of strengthening and relaxation of the network's connections. This interaction appears appropriate in certain regimes because it manages very well to solve different stimuli through the amplification of the differences between distant stimuli. Yet in certain regimes the same mechanism shows some constraints, among which there is the target of our model.

Following the experimental schema of Figure 3, we carried on numerical simulations. The experimental setting is simple enough to generate large amount of data which can be used to estimate model parameters with high accuracy. The model was composed by 20000 nodes and by the same number of synapses; we considered $\alpha = 0.001$, $\gamma = 0.01$, $\theta = 0.1$, $\epsilon = 0.01$, $k_a = 0.1$, $k_x = 0.1$, moreover the length of stimulus has been fixed to 400 time steps. When the stimulus is

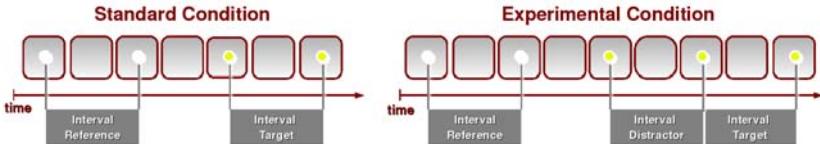


Fig. 3. In the experiment [15] simple visual stimuli were administered separated by interval of several lengths. In the standard condition (*left*) only a target and reference couples of stimuli were presented, while in the experimental condition (*right*) a distractor was administered before the first target stimulus. Interval between distractor and first stimulus has been manipulated in order to elucidate its effects. Finally the esteem of interval between target stimuli, obtained comparing it with the interval between reference stimuli, for both conditions has been compared.

administered the chain is initialized setting the activation value of the first node of the chain to 1. The first node remains at 1 for all the stimulus duration and it is set to the rest value (k_x) when no stimulus is arriving. Control parameters have been varied between 1000 and 9000 time steps. The read-out process of the intervals from the neural chain is composed by two phases. In the first step those neuron which show the greater value of activation with respect to their neighbours are detected and labelled "local maxima". Afterwards the "distance" (e.g. the number of nodes) between those maxima of activations is assumed as the estimate of the target interval. The difference between the target interval estimate in the standard condition and in the experimental condition has been assumed as order parameter of the model.

3 Results

Numerical results are reported in Figure 4(a) and 4(b), on the vertical axis we put the difference between the evaluation of the interval between the target stimuli in presence of distractor, and the one in the control test. Moreover the abscissa and the ordinate of Figure 4(a) show the temporal intervals between the target stimuli (*real gap*) and between the distractor and the first target stimulus (*distractor gap*). The distortion caused by the distractor on the target interval estimate is qualitatively similar to [15]. When the distractor is near the first target stimulus a dilatation of the target interval is detectable with respect to the standard condition. The dilatation of *perceived* time is here caused by the different velocity of propagation of the signals which encode target stimuli. The first target signal propagates more quickly than the second one because it finds still excited synapses by the passage of the distractor stimulus. On the contrary for greater values of the target interval, when the distractor is enough distant from the first stimulus, a compression of the target interval emerges Figure 4(a). Consequently the compression is detectable when the second target signal encounters a medium (i.e. the state of the synapses) more excited than the first one. Noteworthy, see Figure 4(b), there are some appropriate values of the parameters of the model

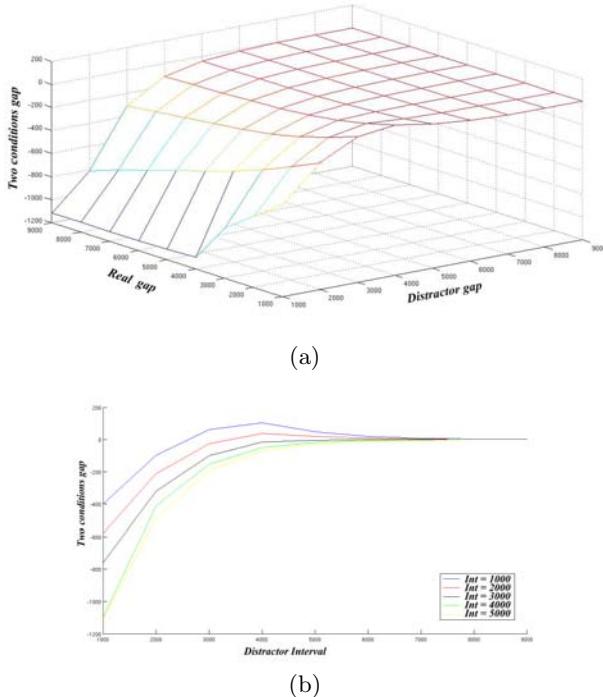


Fig. 4. Difference of target interval estimation (TIE) between reference and experimental conditions, indicated in the ordinate as "Two conditions gap", for different values of real and distractor gap. (a)- Surface in figure shows the effects of *real* and *distractor gap* on the TIE. (b)- Plot shows sections of the surface for different values of the variable *real gap*.

depending on which the model reproduces the experimental findings of Rocca et al [15]. In the model we observe that for extreme values of the distractor gap there is no detectable difference in the evaluation of the temporal interval. It is also interesting to note that for very low values of the distractor gap, i.e., when the distractor is presented immediately before the first target stimulus in temporal terms, the distortion in the precision of the evaluation of the interval between the two condition considerably diverges independently from the interval to evaluate (*real gap*). Finally it is possible to notice an interesting phenomenon which seems to provide confirmation of the qualitative phenomenology observed. In fact, for appropriate values of control parameters (*real and distractor gaps*) we achieve both the phenomenon of temporal compression and temporal dilatation.

4 Extension to Neuron Columns

In our model the spikes travel along a chain of neurons. If one neuron in the chain becomes damaged or dies, the entire chain becomes ineffective. Given that

neurons are dying during human aging, the natural selection has evolved mechanisms to avoid significant loss in information processing ability. It is now known the actual processors of the vision and other processes are cortical columns of neurons. Abeles and followers have proposed that cortical activities are based on two-dimensional neural networks whose connections are organised in a feedforward manner [2]. Localized waves of synchronous spiking activity travel along a sequence neurons organised in layers or pools [18]. If the number of excited neurons in a pool is above a threshold determined by the connectivity between layers, the wave activity initiated at the first pool propagates from one pool to the next, forming a synfire wave. The basic mechanism of transmission relies on a trade-off of excitatory and inhibitory connections between neurons in consecutive pools; each neuron is receiving inputs by many neurons in the previous pool. The firing activity along a chain may propagate in either an asynchronous (i.e. a signal that reaches a threshold firing rate in one pool will cause an elevation in the firing rate in the next pool) or in a synchronous mode (i.e. a synchronous volley in one pool will elicit a synchronous volley in the other after one synaptic delay). Hertz showed that the only viable mode of transmission is the synchronous mode [8]. With these properties in mind, we considered an extension of the model to a single column of neuron and then to multiple columns. In the single column we generated and interconnected randomly sets of neurons, considering both inhibitory ad excitatory synapsis. The multiple column model considers the angle differences between neurons belonging to different pools. Considering single column neurons, we found results qualitatively similar to the single chain case. Work in progress focuses on multiple column case.

5 Conclusion

We present the model that reproduces the qualitative phenomena which characterized the experiment on using a distractor when measuring the visual timing ability. The real effect under scrutiny is here explained by a non linear behaviour of the neuron delay chain activation on the waves propagation velocity. The model could account also for the latency reduction found by Ibbotson et. al [9], explaining it as a consequence of changes occurred in the network activation state. Our model relies on local effects, because of the association of a neuron delay chain with each receptive field of the visual areas. Work in progress focuses on reproduce the real magnitude and time course of the considered phenomena.

Acknowledgments. We thank our colleagues of Pisa Vision Lab for suggestions. This work is financed by the EU 6 Framework Programme Project: Measuring and Modelling Relativistic-Like Effects in Brain and NCSs.

References

1. Abarbanel, H., Gibb, L., Huerta, R., Rabinovich, M.: Biophysical model of synaptic plasticity dynamics. *Biol. Cybern.* 89, 214–226 (2003)
2. Abeles, M.: Local Cortical Circuits. Springer, Berlin (1982)

3. Buonomano, D.V.: Decoding Temporal Information: A Model Based on Short-Term Synaptic Plasticity. *The Journal of Neuroscience* 20(3), 1129–1141 (2000)
4. Burr, D.: Eye Movements: Keeping Vision Stable. *The Current Biology* 14, 195–197 (2004)
5. Burr, D., Morrone, C.: Perception: Transient Disruptions to Neural Space Time. *The Current Biology* 16(19), 848 (2006)
6. Burr, D., Morrone, C.: Time Perception: Space time in the Brain. *Current Biology* 16(5), 171–173 (2006)
7. Dittman, J.S., Kreitzer, A.C., Regehr, W.G.: Interplay between facilitation, depression, and residual calcium at three presynaptic terminals. *Journal of Neuroscience* 20, 1374–1385 (2007)
8. Hertz, J., Prugel-Bennett, A.: Learning short synfire chains by selforganization. *Network* 7, 357–363 (1996)
9. Ibbotson, M., Crowder, N., Price, N.: Neural basis of time changes during saccades. *Current Biology* 16(19), 834–836 (2005)
10. Johnston, A., Arnold, D.H., Nishida, S.: Spatially Localized Distortions of Event time. *Current Biology* 16, 472–479 (2006)
11. Karmarkar, U., Buonomano, D.: Timing in the Absence of Clocks: Encoding time in Neural Network States. *Neuron* 53(3), 427–438 (2007)
12. Lewis, P.A., Walsh, V.: Time perception: components of the brain's clock. *Current Biology* 15(10), 389–391 (2005)
13. Guazzini, A., Lio, P., Conti, M., Passarella, A.: Complex network modeling of saccades control, *Frontiers in Neuroinformatics*. In: *Neuroinformatics 2008* (2008), doi:10.3389/conf.neuro.11.2008.01.039
14. Meck, W.H.: Neuropsychology of timing and time perception. *Brain Cognition* 58(1), 1–8 (2005)
15. Rocca, E.D., Burr, D.C.: Large contrast effects in the discrimination of short temporal intervals. *Perception* 36 ECVP (2007)
16. Ross, J., Burr, D., Morrone, C.: Suppression of the magnocellular pathway during saccades. *Behavioural Brain Research* 80, 1–8 (1996)
17. Ross, J., Morrone, C., Goldberg, E., Burr, D.: Changes in visual perception at the time of saccades. *TRENDS in Neurosciences* 16, 472–479 (2001)
18. Wennekers, T., Palm, G.: Controlling the speed of synfire chains. In: Malsburg, C., Seelen, W., Vorbruggen, J., Sendhoff, B. (eds.) *ICANN 1996. LNCS*, vol. 1112. Springer, Heidelberg (1996)

Review of Neuron Types in the Retina: Information Models for Neuroengineering

German D. Valderrama-Gonzalez, T.M. McGinnity,
Liam Maguire, and Qing Xiang Wu

Intelligent Systems Research Centre, School of Computing and Intelligent Systems,
University of Ulster at Magee Campus, Derry, BT48 7JL, Northern Ireland, UK

<http://isrc.ulster.ac.uk/>

Abstract. Powerful information processing functions are performed in the mammalian retina in which basic units are different types of neurons. This paper presents the types of neurons and their roles in the visual processing system. The aim is to review the principles of how an artificial visual system could be constructed based on a comprehensive understanding of biological systems.

Keywords: Retina, cell types, neuroengineering, computational neuroscience, artificial visual systems.

1 Introduction

A challenging goal for engineers and computer scientists is to apply knowledge from biological research to create human-like intelligent artificial visual systems (AVS) [1, 2]. In order to achieve this goal, it is necessary to dissect the key principles that underpin biological vision [3, 4, 5, 6]. The focus of this paper is to review the different types of neurons in the retina and their circuitry to process information. A summary of anatomy and physiology is presented along with some implications for modelling.

2 A Review of the Biological Retina Cell Types

Histologically, the retina is divided in 5 layers: (i) outer nuclear layer which contains the nucleus of the photoreceptors, (ii) the outer plexiform or interplexiform layer (IPL) where the synapses between photoreceptors, horizontals and bipolars take place, (iii) the inner nuclear IPL layer which contains the cells bodies of horizontal, bipolar and amacrine cells, collectively called interneurons, (iv) the inner IPL where the synapses between amacrine, bipolar and retinal ganglion cells (RGCs) take place and finally (v) the ganglion cell layer which contains the soma of the RGCs. The retina can be compared to a neural network (NN) of three layers (Fig. 1a) [7, 8] where the second and fourth histological layers are just connections. The outer nuclear layer where the photoreceptors are located corresponds to the NN input layer. The hidden layer of a NN is analogous to the

inner nuclear IPL that contains the nucleus of the interneurons. The ganglion cell layer is analogous to the NN output layer.

Photoreceptors are cells that respond to light by graded changes in the membrane potential. Photoreceptors have an hexagonal shape and an ON and OFF receptive field (RF) organisation [3]. There are two kinds of photoreceptors in the human retina: cones and rods. **Rods** are responsible for night vision, they detect dim light present at dusk or at night (moonlight). The rod system is highly sensitive to light, but is achromatic. **Cones** are responsible for day vision, they provide better spatial and temporal resolution, and mediate colour vision. Cones are densely packed in the centre of the retina, the fovea, thus providing the retina with a high resolution area in the centre; the surround of the retina has equilibrium but the periphery is vastly dominated by rods [4]. On Figure 1, for visual simplification, cones are shown contacting a single bipolar cell; in reality they contact several bipolars of the same type and viceversa. The (L)ong or red cone and the (M)iddle wavelength-sensitive or green cone (Fig. 1a) transmit “red-green colour opponency” signals to the midget ganglion cell (P-RGC) [9]. The short wavelength cone termed S or ‘blue cone’ has its own circuitry system (Fig. 1c). Chromatic information in a single channel tells the brain where along the spectrum from blue to red-green the stimulus lies (i.e. “blue-yellow colour opponency” signal).

Horizontal cell response is a function of the average intensity of stimulation within its RF. Horizontals enhance the concentric sensitivity of bipolar and RGCs, exciting or inhibiting surrounding photoreceptors. In mammals [9,10] there are two types of horizontals: H1 and H2. L- and M-cones relay and receive feedback (inhibition) from H1 horizontals (Fig. 1d). H1 cells sum the L- and M-cone inputs to produce a “L-M-cone-contrast-gain” [9]. H2-rods are far from the horizontal soma, making H2 cells electronically isolated from the cone’s feedback system. Thus, rods have an independent feedback system driven by and fed by rods [10].

Bipolar cells participate in the straight forward flux of visual information or direct pathway as well as on the indirect or lateral pathway. If the direct pathway is excitatory, the indirect pathway is inhibitory and vice versa [4]. Both pathways are integrated by the bipolar to generate a centre/surround or ‘Mexican hat’ RF. There are two size types of bipolars; the larger type innervates the rods (‘rod-bipolars’) while the smaller type is connected to the cones (‘cone-bipolars’) [5]. The mammalian retina contains 9-11 types of cone-driven bipolars [5,11] which subdivide into transient and sustained (Fig. 1b). Transient bipolar cells (b1) respond ON- and OFF- to high frequency light present anywhere in their RF. Sustained bipolars (b2/b3) which respond to low frequency light are also split on ON-sustained and OFF-sustained. This subdivision filters the signal into its temporal components, thus adding temporal filter behaviour to the retina [12].

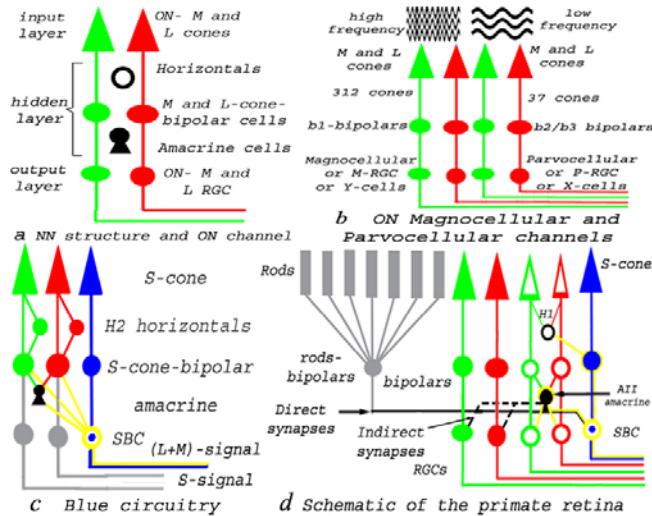


Fig. 1. This diagram shows the basic information processing channels in the primate retina and much detail is omitted. (a) The retina can be compared to a NN. Comparative chromatic information along the red to green spectrum is gathered in the (L-M)-channel. (b) ON version of the magnocellular and parvocellular channels. (c) Koniocellular pathway. (d) Rod circuitry. See text for details.

Amacrine cells: There are 20 morphologically different types of amacrines that use at least 8 different neurotransmitters. Some amacrines have similar functions to horizontals: they mediate antagonist inputs to RGCs from bipolars in the RGC's surround. Others have been implicated in shaping the complex RF properties of specific classes of RGCs. Amacrines also have two different types of response: transient and sustained. Rods and cones connect in parallel to separate bipolars, forming a separate cone and a rod circuitry. Rods' circuitry are simpler than cones (Fig. 1d) [5]. At least two different pathways convey rod signals to RGCs in the moderately and extremely dark-adapted eye [4]. Hundreds of rods convey information to a single rod-driven bipolar synapsing on AII amacrine cell which give their output (a) directly to OFF-centre-RGCs and (b) indirectly to ON-centre-RGCs via cone-bipolars. Thus, the rod pathway attains access to the highly structured circuitry of the cone pathway, i.e. a directionally selective RGC maintains its function in dim light [4,5].

Ganglion cells receive their input from bipolar and amacrine cells. They transform this information into APs that are the output of the retina. This is the first class of neurons in the visual pathways which actually uses APs to transmit information. Visual information includes light intensity (driven by rods) and colour information (driven by cones) [4]. Their soma is located in the retina but

their long axons form the optic nerve that ends at different nuclei in the thalamus, like the LGN. The neurons whose soma is at the LGN and axon travel to V1 in the visual cortex and are also called ganglion cells. A long-held principle is that each ganglion cell type projects information to the brain in distinct parallel pathways; however, not all ganglion types leads to a different pathway [5,13]. Three pathways are well known:

Magnocellular pathway: Some L-M-cones respond more vigorously to high frequency light. H1-horizontals biased those transient L-M cones. They both gather information to b1-bipolars and transient amacrices, which in turn pass it to M-RGC. M-RGCs are also called parasol cells, Y-ganglion cells in monkeys and α -cells in cats [4,13,14]. M-RGCs are large neurons that have large concentric RFs which respond in a non-linear way with a transient response to temporal frequency modulation. M-RGCs mainly get input from the periphery of the retina and have a high degree of convergence. A single M-RGC receives input from approximately 100 rod-bipolars which in turn receive input from as many as 1500 rods and 8-10 cones-bipolars type b1 which in turn receive input from 312 cones [4,5]. This high convergence enhances sensitivity of the magnocellular channel. The high sensitivity (to light) of the rods, added to the high convergence provides a night vision channel highly responsive to movement. M-RGCs project to the first and second layers of the LGN. There are two types of neurons on this channel: ON and OFF.

Parvocellular pathway: Some other L-M-cones are more sensitive to low frequency light. They feed b2/b3-bipolars and sustained amacrices, which in turn biases P-RGC. P-RGCs are also called β -ganglion cells in cats, midget ganglion cells or X-ganglion cells in monkeys [4,13,14]. P-RGCs are small, having a small linearly summing RF with a sustained response. P-RGCs are found nearer the fovea and are concerned with high resolution vision. In the fovea, P-RGCs constitute about 90% of all the RGCs whereas in the periphery they constitute only 40-45% [5]. P-RGCs end at the Parvocellular (3rd - 6th) layers of the LGN. P-RGC also comes in ON and OFF versions.

Koniocellular pathway: S-cones provide their ON-signal to their own specialised ([S-ON/(L+M)-OFF]) bipolar cells. The (L+M)-OFF signal is provided with a delay by surrounding cones that had been contacted by H2-horizontals [9,16]. Specialised bipolars relay to a dedicated class of RGCs, the blue-ON/ yellow-OFF or [S-ON/(L+M)-OFF]-RGC or small bistratified cell (SBC) [13,15,17]. Nonetheless, there is no agreement in the literature whether the OFF-(L+M) signal circuitry is cones-H2-bipolars-SBC [9,13,17] or cones-bipolars-amacrine-SBC [18] or a mixture [19] (Fig. IIc). Yet, another alternative is that H1-horizontals contact M-bipolars providing the (L+M)-OFF signal directly to SBC [9].

Table 1. Retinal ganglion neuron types. LGN = Lateral geniculate nucleus; SC = Superior colliculus; AOS = Accessory optic system; SQ = Suprachiasmatic nucleus.

Name	Subtype(s)	Function or characteristic	
Magnocellular, Parasol, α and Y cells [1][2][3][4]	ON and OFF	temporal frequency modulation	LGN (Magnocellular)
Parvocellular, Midget, β and X cells [1][2][3][4]	ON and OFF	high visual resolution	LGN (Parvocellular)
SBC, S-ON/(M+L)-OFF [3][4][5][6][7]	blue-ON/yellow-OFF	spectral opponency signal [6]	LGN (Koniocellular) [5]
Large bistratified [20][21], G17 [22]	blue-ON/yellow-OFF [9]	different temporal properties than SBC [2]	LGN [1]
Broad thorny [20][21], T group [11], G5 [22]	ON and OFF	phasic response to light [23]	LGN [1] and SC [21]
Bistratified [21], broad wavy [20], S group [11]			SC [2]
Large very sparse [20][21], PT	sustained-ON [23]	mediates light reflex and controls pupillary size [5][24]	prectectal area and SC [5][24]
Large sparse [13][20][21], G3 [22], Gamma [11]	inner and outer		prectectum [11]
Giant sparse cell [13][20][21], G3 [22], Gamma [11]			prectectum [11] and AOS [5]
Multi-tufted [20], broad thorny [21], PT dense [11]	ON and OFF	direction selective [25]	
Diffuse thorny [20], dense diffuse [21], G5 [22]		Bistratified	
Diffuse smooth [20], G22 [22]		Bistratified	
Monostratified large sparser [20][21], G20 [22]	blue-OFF	spectral opponency signal	LGN (Parvocellular)
Large moderate [20][21], G21 [22]	ON	direction-selective	AOS [5]
Narrow thorny [20][21], G8,G16 [22]		local edge detector [26]	
Monostratified large dense [20][21], maze [11]			SC [11]
Monostratified large radiate [20], large sparse [21]	ON or OFF	non-linear [27]	LGN [1]
		involved in control of circadian rhythms [24]	SQ [2]

Extracellular recordings of APs from the optic nerve allow fairly easy visualisation of RGC's activity. Thanks to systematic anatomical studies (visually guided microinjection) and electrophysiology research in monkeys and humans 17-22 different types of RGC have been identified, as summarised in Table 1

3 Implications for Modelling

From the above review it is clear that visual information is processed in the retina by different types of neurons. Understanding the neurons circuitry and computations leads to understanding the information processing in the retina. The state-of the art of computational neuroscience of vision is that only a few types of neurons of the visual system are modelled. Therefore, we present a set of issues for simulation of an early stage of an AVS as follows: **(1)** What difficulties will we encounter when a visual computational model is used with multiple neuron types? Several issues can arise here, **(a)** Photoreceptors in biological systems are hexagonal, therefore a biologically inspired computational model should be arranged in a hexagonal grid [8]. However, there are no commercially available cameras that takes “images” using hexagonal pixels or “ixels” rather than conventional square pixels. See [28] for a review on hexagonal image processing. **(b)** Also, the computational power required to do this processing in a biologically inspired way is huge and was not possible a decade ago. **(c)** In spite of great scientifical advances, our knowledge of the visual system neuron types, their connections and computations remain unclear. **(2)** Based on what is found on biology it is useful to assume that AVS would be more powerful if more types of neurons were considered in the models. For example, this review shows that the retina contains two forms of photosensitive neurons, i.e. rods and cones. Three types of cones have primary sensitivity of red, green, or blue light. By using three types of cones and rods for modelling of the visual system the colour processing mechanism of the retina in a wide variety of light conditions may be implemented in AVS. This mechanism is able to combine the gray-scale and RGB signals instead of only gray scale in current AVS models. **(3)** How can we use multiple neuron types in one visual system model? This can be achieved by oversampling the image and using a highly organised sampling architecture based on RFs and by implementing different channels each with particular neuron types that extract a specific feature from the image.

4 Architectural Model

Based on biological findings [6] it is possible to create early stage modelling of an AVS (Fig. 2). The initial stage of development of this model is the retina (encircled yellow) which is the focus of this paper. As shown in the diagram, the magnocellular pathway contributes to both the ‘where’ and the ‘what’ pathways where as the parvocellular mainly contributes to the ‘what’ pathway. On an early stage of modelling blue-cone signal can be transmitted by the magnocellular and parvocellular pathways. Thus, leaving us with four channels: ON and OFF

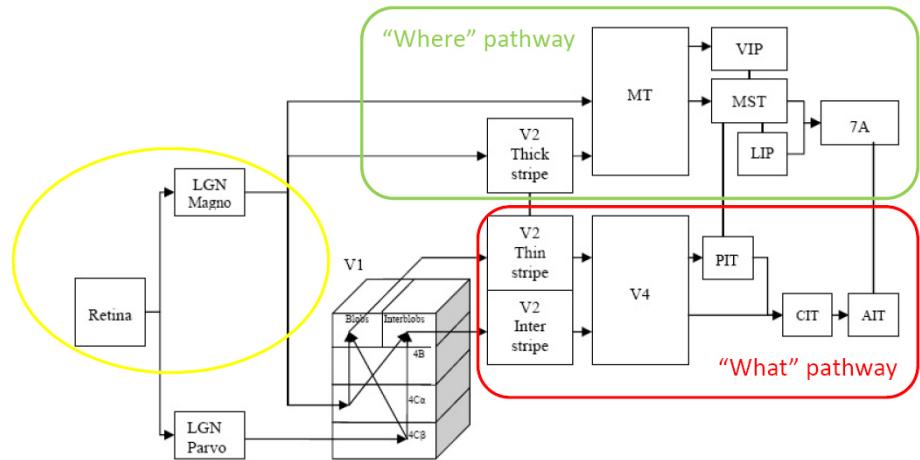


Fig. 2. Hierarchical organization of the overall proposed artificial visual system model. The Where pathway that performs motion detection and the What pathway that performs object recognition. Modified from [6].

magnocellular and ON and OFF parvocellular. Each of those channels relay red (L), green (M), blue (S) and grey (rod) type of information. As an example the L-(ON and OFF)-magnocellular channels are described next. The L-ON-Magnocellular-RGC processes an ON version of high frequency red light within its RF (Fig. 1b and Fig. 3a), and a L-OFF-Magnocellular-RGC processes the OFF version of the same signal (Fig. 1d and Fig. 3c). Neighbouring RGC of the same type convey to a low hierarchy complex cell in the brain with a complex RF (Fig. 3b,d,e). Sequentially this cell conveys information to a higher level cell that compiles information regarding high frequency red light in the entire image (Fig. 3f). Each RGC type processes specific information coming from a particular area (i.e. RF) in the visual field (i.e. image).

Now, how to combine those theoretical channels with a practical architectural model? An easy way to achieve this is to apply the following architectural model of the retina to each colour type of information. In this hypothetical architectural model the centre of each simple-RF is formed by 7 hexagonal ‘hixels’ and the surround is formed by two hexagonal rings of ‘hixels’ (Fig. 3a). ON-complex-RFs are composed of 7 hexagonal ON-simple-RFs (Fig. 3c). For the ON-Magnocellular channel when arranged to constitute the complex-RF, each simple-RF has six overlapping ‘hixels’; thus, overlapping a single (strategically located) ‘hixel’ with each neighbour. Also note that there are ‘hixels’ that are not-included on the ON-Magnocellular-complex-RF arrangement (Fig. 3c). The non-including ‘hixels’ of the ON-Magnocellular complex-RFs are the centre of the OFF-Magnocellular complex-RFs (Fig. 3d-f). As this ‘hixel’ was not included on the ON representation this means that it is not the same RF with inverted sign but rather another different channel. For the surrounding of the

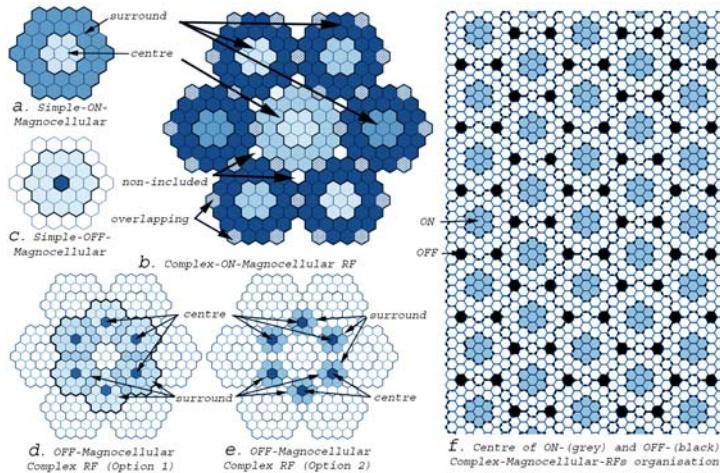


Fig. 3. Hypothetical model of Magnocellular simple- and complex-RFs. (a) ON-Simple and (b) ON-Complex Magnocellular-RF. The centre of this complex-RF is a simple-RF and the surround is drawn by six simple neighbouring RFs. (c) OFF-Simple (for option 1) and (d-e) OFF-Complex Magnocellular-RF, see text. (f) Only the centres of ON-(blue) and OFF-(black) Complex-Magnocellular-RFs organisation are shown.

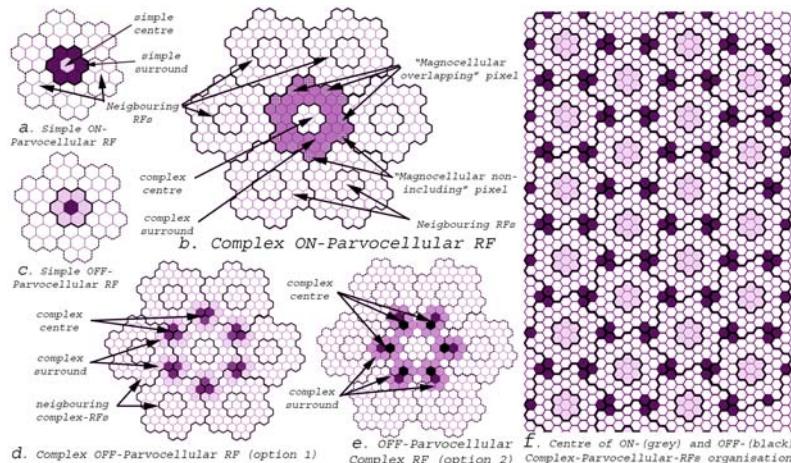


Fig. 4. Early stage model of simple and complex-Parvocellular-RFs. (a)ON-simple and (b) ON-Complex-Parvocellular-RF. (c) OFF- Simple-Parvocellular-RF. (d) The magnocellular's non-included-hixel is the centre of the OFF-Complex-Parvocellular channel. (e) The magnocellular's overlapping-pixel is the centre of the OFF-Complex-Parvocellular channel. (f) Centre of ON-(purple) and OFF-(black) Complex-Parvocellular-RFs' organisation.

OFF-Magnocellular complex-RF, there are two options whether an overlap is beneficial (option 1) or not (option 2). The first option is a big surrounding area formed by two rings of ‘ixels’ so that there is a continuity of the surrounding area (Fig. 3d). If this continuity that implies overlapping was undesirable an smaller surrounding area formed by just one ‘ixel’ ring would be more appropriated (Fig. 3e). In both cases, the centre of the ON-Magnocellular complex-RFs is respected (Fig. 3b). Thus, despite the overlapping in the surround on the selection of pixels (photoreceptors) there is no overlapping in the organisation of complex-RF. The centre of both ON- and OFF- Magnocellular channels are shown in Figure 3f. Moreover, in this hypothetical model as in biology, the Parvocellular simple-RFs are smaller than their Magnocellular counterpart. For that reason the centre is just a ‘ixel’ and the surround is just a ring of six ‘ixels’ (Fig. 4a,c). The centre of the ON-Parvocellular complex RF, and the surround is shaped by six neighbouring simple-RFs (Fig. 4b). As for the OFF system, there are also two options. The first option is that the OFF-centre is outlined by the “Magnocellular’s non-included” hixel (Fig. 4d). The second option is that the OFF-centre is delineated by the “Magnocellular’s overlapping” hixel (Fig. 4e). Both ON and OFF centres of the Parvocellular channel are displayed on Figure 4f. As presented here, the Parvocellular system has not overlapping at all.

5 Conclusions

The mammalian retina contains different neuron types and subtypes, each with a different function. Almost each RGC type has its own circuitry or pathway that processes a specific visual response property along the visual field. In each of those individual channels information gathered at photoreceptor level is relayed in purpose-specific cells that leads to specific feature extraction of the visual information. Consequently, the retina output is composed of multiple parallel specific feature representations of the same image. Each of those individual specific pathways travels to a specific area in the visual cortex for higher processing. Thus, a promising way to model the retina is to model the individual channels in which each channel processes a specific feature. Therefore, a potential computational model of vision that solves those issues leads to future AVSs.

References

1. Bharath, A., Petrou, M.: Next Generation Artificial Vision Systems Reverse Engineering the Human Visual System. Artech House (2008)
2. Miikkulainen, R., Bednar, J.A., Choe, Y., Sirosh, J.: Computational Maps in the Visual Cortex. Springer, Heidelberg (2005)
3. Hubel, D.H.: Eye, Brain and Vision. Scientific American, New York (1988)
4. Rodieck, R.W.: The First Steps in Seeing. Sinauer associates, Sunderland (1998)
5. Masland, R.H.: The fundamental plan of the retina. Nat. Neurosci. 4, 877–886 (2001)
6. Merigan, W.H.: How parallel are the primate visual pathways? Annu. Rev. Neurosci. 16, 369–402 (1993)

7. Lehky, S.R., Sejrtowski, T.J.: Network model of shape from shading: Neural function arises from both receptive and projective fields. *Nature* 333, 452–454 (1999)
8. Huang, C., Lin, C.: Bio-Inspired Computer Fovea Model Based on Hexagonal-Type Cellular Neural Network. *IEEE Transactions on circuits and systems part 1 regular papers* 54, 35–47 (2007)
9. Dacey, D.M.: Parallel Pathways for Spectral Coding in Primate Retina. *Annual Reviews in Neuroscience* 23, 743–775 (2000)
10. Wässle, H., Dacey, D.M., Haun, T., Haverkamp, S., Grünert, U., Boycott, B.B.: The mosaic of horizontal cells in the macaque monkey retina: With a comment on biplexiform ganglion cells. *Vis. Neurosci.* 17, 591–608 (2000)
11. Rodieck, R.W., Watanabe, M.: Survey of the morphology of the macaque retinal ganglion cells that project to the pretectum, superior colliculus and parvocellular laminae of the lateral geniculate nucleus. *J. Comp. Neurol.* 338, 289–303 (1993)
12. DeVries, S.H.: Bipolar Cells Use Kainate and AMPA Receptors to Filter Visual Information into Separate Channels. *Neuron* 28, 847–856 (2000)
13. Field, G.D., Chichilnisky, E.J.: Information processing in the primate retina: circuitry and coding. *Annu. Rev. Neurosci.* 30, 1–30 (2007)
14. Wässle, H., Boycott, B.B.: Functional architecture of the mammalian retina. *Physiological Reviews* 71, 447–470 (1991)
15. Dacey, D.M., Lee, B.B.: The ‘blue-on’ opponent pathway in primate retina originates from a distinct bistratified ganglion cell type. *Nature* 367, 731–735 (1994)
16. Dacey, D., Peterson, B., Robinson, F.: Identification of an S-cone Opponent OFF Pathway in the Macaque Monkey Retina: Morphology, Physiology and Possible Circuitry. *Invest. Ophthalmol. Vis. Sci.* 43, 2983 (2002)
17. Kouyama, N., Marshak, D.: Bipolar cells specific for blue cones in the macaque retina. *Journal of Neuroscience* 12, 1233–1252 (1992)
18. Lukasiewicz, P.D.: Synaptic mechanisms that shape visual signaling at the inner retina. *Prog. Brain Res.* 147, 205–218 (2005)
19. Ichinose, T., Lukasiewicz, P.D.: Inner and outer retinal pathways both contribute to surround inhibition of salamander ganglion cells. *J. Physiol.* 565, 517–535 (2005)
20. Yamada, E.S., Bordt, A.S., Marshak, D.W.: Wide-field ganglion cells in macaque retinas. *Vis. Neurosci.* 22, 383–393 (2005)
21. Dacey, D.M., Peterson, B.B., Robinson, F.R., Gamlin, P.D.: Fireworks in the Primate Retina In Vitro Photodynamics Reveals Diverse LGN-Projecting Ganglion Cell Types. *Neuron* 37, 15–27 (2003)
22. Kolb, H., Linberg, K.A., Fisher, S.K.: Neurons of the human retina: A Golgi study. *J. Comp. Neurol.* 318, 147–187 (1992)
23. Isayama, T., Berson, D.M., Pu, M.: Theta ganglion cell type of the cat retina. *J. Comp. Neurol.* 417, 32–48 (2000)
24. Moore, R.Y., Speh, J.C., Card, J.P.: The retinohypothalamic tract originates from a distinct subset of retinal ganglion cells. *J. Comp. Neurol.* 352, 351–366 (1995)
25. Pu, M., Berson, D.M., Pan, T.: Structure and function of retinal ganglion cells innervating the cat’s geniculate wing: an *in vitro* study. *Journal of Neuroscience* 14, 4338–4358 (1994)
26. Amthor, F., Takahashi, E., Oyster, C.: Morphologies of rabbit retinal ganglion cells with complex receptive fields. *Journal of Comparative Neurology* 280, 97–121 (1989)
27. Kaplan, E., Shapley, R.M.: X and Y cells in the lateral geniculate nucleus of macaque monkeys. *The Journal of Physiology* 330, 125–143 (1982)
28. Middleton, L., Sivaswamy, J.: *Hexagonal Image Processing: A Practical Approach*. Springer, Heidelberg (2005)

Brain Electric Microstate and Perception of Simultaneously Audiovisual Presentation

Wichian Sittiprapaporn¹ and Jun Soo Kwon^{2,3,4}

¹ College of Music, Mahidol University, Salaya, Nakhonpathom, Thailand

² Department of Psychiatry, Seoul National University College of Medicine,
Seoul, Korea

³ Clinical Research Center, Seoul National University Hospital, Seoul, Korea

⁴ Brain Korea 21 Human Life Science, Seoul National University, Seoul, Korea
Tel.: +82-2-2072-2972; Fax: +82-2-747-9063

kwonjs@snu.ac.kr

Abstract. Associations between picture and sound form the basis of reading. Learning the correspondences between them is a crucial step in reading acquisition. This study was designed to investigate whether task-related processing of audio and visual features was independent or task-related processing in one modality might influence the processing of the other. The present study employed simultaneous audio-visual stimulus in the oddball paradigm to re-examine the effects of attention on audio, visual and audio-visual perception in the non-musician brain. Electroencephalographic (EEG) was recorded from 28 normal participants. None of them had more than three years of formal musical training and none had any musical training within the past five years. Chinese and Korean subjects were presented with tones (auditory: A), pictures (visual: V), and simultaneous tones and pictures (audio-visual: AV). The neural basis of this interaction was investigated by subtracting the event-related potentials (ERPs) to the A and the V stimuli alone from the ERP to the combined AV stimuli (i.e. interaction = AV - (A+V)). The Korean group showed larger mean interaction amplitude and longer in time than the Chinese group. This reveals that experience influences the early cortical automatic processing of linguistically relevant suprasegmental pitch contour. These results suggest that efficient processing of associations between pictures and sounds relies on neural mechanisms similar to those naturally evolved for integrating audiovisual perception.

Keywords: Brain, Audiovisual perception, Cognition, Event-related potential (ERP), Microstate.

1 Introduction

Theoretically, the human central auditory system has a remarkable ability to establish memory traces for invariant features in the acoustic environment such as music and speech sounds in order to correct the interpretation of natural acoustic sound heard. Even when no conscious attention is paid to the surrounding sound,

changes in their regularity can cause the listener to redirect his or her attention toward the sounds [1]. Event-related potential (ERP) recordings have brought new insight to the neuronal events behind auditory change detection in audition. ERPs components reflect the conscious detection of a physical, semantic, or syntactic deviation from the expected sounds [1]. The ERP recordings thus allow one to probe the neural processes preceding the involvement of the attentional mechanisms. For instances, ERPs have been recorded that reflect memory traces representing sounds composed of several simultaneous or successive tonal elements [2-4].

Mismatch negativity (MMN) component of ERP is elicited in the auditory cortex when incoming sounds are automatically detected as deviating from a neural representation of acoustic regularities. It is mainly generated in the auditory cortex [5] occurring between 100 to 250 ms and thus long been regarded as specific to the auditory modality [6-7]. The automatic change-detection system in the human brain as reflected by the MMN requires the storage of the previous state of the acoustic environment for detecting an incoming deviating sound [6,8]. MMN implies the existence of an auditory sensory memory that stores a neural representation of a standard against which any incoming auditory input is compared [9]. In the auditory modality, MMN is an automatic process which occurs even when the subject's attention is focused away from the evoking stimuli [6]. Its onset normally begins before the N2b-P3 complex which occurs when attention is directed to the stimuli. The duration of MMN varies with the nature of the stimulus deviance but it invariably overlaps N2b when the latter is present [10]. Previous study [11] has stated that the automatic detection of stimulus change plays a part in directing attention to events of biological importance. If this is the case, one would expect a similar mechanism to operate in the visual modality. Even though MMN had not mentioned to be appeared in the visual modality [11], several studies have shown that visual stimuli deviating from repetitive visual standards can also elicit a visual analogue of the MMN in the same latency range. This visual MMN seems to be mainly generated in occipital areas [12-13] with possibly a more anterior positive component [14-15]. In addition, Cammann's study [16] showed a widely distributed MMN change between 150 and 350 ms, with a parietal maximum suggesting that this MMN may occur in the visual modality. Recently, Pazo-Alvarez et al. [17] reviewed several previous reports to provide convincing evidence for the existence of this visual MMN. Moreover, cross-modal attention studies clearly showed that deviant visual stimuli elicited MMN, largest over the inferior temporal cortex. This visual MMN increased in amplitude with attention, but it was also evident during inattention [18-19].

It is well-known that auditory signals can be differentiated by a variety of factors, including spectral and temporal information. It is important to recognize that music and speech sounds differ in the way they exploit spectral and temporal cues. The present work focuses on three related issues: How is the brain's music and speech sounds processing represented in brain electric potential fields, how are the music and speech sounds with different pitch contours distinction

embedded in this processing and which implications do these finding have for theories of the human brain processing. The aim of the present study thus was to study the human brain's audiovisual integration mechanisms for pictures, i.e., for stimuli that have been previously associated through learning. The subjects received auditory, visual, and audiovisual and were required to identify them, regardless of stimulus modality. Audiovisual stimuli included matching pictures, in which the auditory and visual stimuli corresponded to each other based on previous experience, and nonmatching (randomly paired) pictures. Meaningless auditory, visual, and audiovisual control stimuli were presented as well. The brain activations were detected with electroencephalogram (EEG), which is well suited for noninvasive identification of cortical activity and its accurate temporal dynamics. This research project will elucidate neural mechanisms involved in involuntary or preattentive discrimination of Chinese phonological units: tones and pitch. By using native and non-native speakers of Chinese, we will be able to determine the extent to which these phonetic processes are sensitive to experience. This study also explored the effects of the relatedness of pictures and speech sounds. Additionally, the goal of the present study was designed to re-examine the effects of attention on MMN in audio, visual and audio-visual dimensions by employing an oddball paradigm. Attentional ERP components were analyzed in a situation where target stimuli were combinations of both audio and visual features. Interactive processing of stimulus features would then be indicated by the absence, reduction or early termination of the attention-related components [20] as a function of processing of the other feature. If visual-specific components are evoked by visual deviances, then the present audio-visual paradigm will help to separate them from the effect of visual information on the audio-specific MMN process by facilitating the focus of attention on audio and visual MMNs elicited with bimodal features.

2 Material and Methods

2.1 Subjects

In this study, fourteen right-handed native speakers of Mandarin Chinese, and fourteen right-handed native speakers of Korean, participated in the ERP experiment. Native speakers of Mandarin Chinese had no Korean instruction before the age of 11, and vice versa. Both groups had normal hearing sensitivity and corrected to normal vision (self reported). None of the participants had more than three years of formal musical training and none had any musical training within the past five years. All participants had no history of neurological or psychiatric history. After a complete description of the intended study, written informed consent was obtained. The participants were paid for their participation.

2.2 Stimuli and Procedure

Stimuli consisted of a set of three synthesized Mandarin Chinese tones that were distinguished minimally by pitch contour: [T1]; [T2]; [T3] and presented in an

oddball paradigm. The duration of the stimuli was 300 ms. The audio-visual experiment included four stimuli: congruent [T1] (acoustic [T1] + visual [T1]), congruent [T2] (acoustic [T2] + visual [T2]), incongruent [T1] (acoustic [T1] + visual [T2]) and congruent [T4] (acoustic [T4] + visual [T4]). The auditory and visual experiments included only the acoustic and the visual parts of these stimuli, respectively. The stimulus onset asynchrony was 1300 ms. Stimulus sequences consisted of frequent ($P = 0.60$) congruent [T1] stimuli and congruent ($P = 0.15$) and incongruent ($P = 0.15$) [T2] stimuli. Congruent [T4] stimuli were presented as target ($P = 0.10$) to be able to check that subjects were attending the stimuli (see Fig. 1). The stimulus sequences presentation was controlled by the stimulus system (STIM2, Neurosoft, Inc. Sterling, USA). While visual stimuli were presented on the computer screen, acoustic/audio stimuli were delivered binaurally to the participants through plastic tubes and earpieces. Sound density was adjusted to be 85 dB above the participant's hearing threshold. Subjects sat in an electrically shielded and soundproofed room with the response buttons under their hands. The subject had to press the button on the response pad when the target was presented and ignore any other types of stimuli.

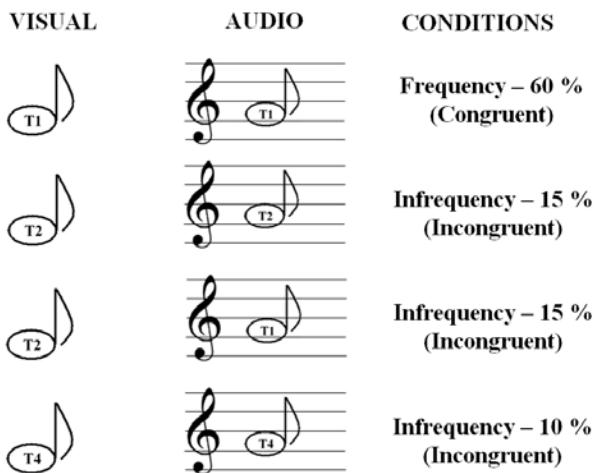


Fig. 1. Illustration of the experimental conditions

2.3 Electroencephalographic Data Processing

EEG data were collected in an electrically and acoustically shield room at the Laboratory of the Clinical Research Institute, Clinical Cognitive Neurosciences Center, Seoul National University Hospital, Korea. EEG was recorded with a Quick-Cap equipped with 64 channels according to the international 10-20 system using Scan system (Scan 4.3, Neurosoft, Inc. Sterling, USA). Reference electrode was at mastoids. The signals were bandpass filtered at 0.05-100 Hz and digitized at 1000 Hz. The impedance of the electrode was below $5\text{ k}\Omega$. Eye

movements were monitored with two EOG electrodes. Four electrodes monitored horizontal and vertical eye movements for off-line artifact rejection. Vertical and horizontal electro-oculogram (EOG) was recorded by electrodes situated above and below the left eye, and on the outer canthi of both eyes, respectively. Epochs with EEG or EOG with a large ($> 100 \mu V$) amplitude were automatically rejected. The artifact-free epochs were filtered at 0.1-15 Hz, baseline corrected and averaged.

2.4 Data Analysis

EEG was segmented into 1,000 ms epochs, including the 100 ms pre-stimulus period. The average waveforms obtained from the standard, deviant and target stimuli were digitally filtered by a 0.1 - 15 Hz band-pass filter and finally baseline-corrected. The N1 that was elicited at approximately 100 ms after the onset of auditory stimulus, was visually inspected from waveform of standard and deviant stimulus. Cross-modal interaction was investigated by subtracting the ERPs to the auditory (A) and the visual (V) stimuli alone from the ERP to the combined audio-visual (AV) stimuli (i.e. interaction = AV - (A+V) and was identified as the peak voltage between 100-250 ms after stimulus onset in the subtracted waveform. By using a peak-detection algorithm, the negative peak was identified in the AV - (A+V) difference waveform between 100 - 250 ms.

2.5 Data Pre-processing and Feature Extraction

For each subject, mean ERP map series for the summed of auditory and visual, AV interaction and audio-visual stimuli were computed over the 6 blocks where each block was weighted by the number of averaged sweeps that it consisted of. The grand mean map series over subjects and conditions was then computed. For all mean ERP map series, the locations of the centroid of each map were computed [21]. Centroids are the points of gravity of the positive and the negative areas of an average reference-referred map.

2.6 Assessment of Changes of Spatial Map Configuration and Map Landscape

For the analysis of changes of spatial map configuration, the curves of the centroid locations over time were averaged over subjects. As the goal was to investigate whether there are periods of stable spatial map configuration in the data, methods for space based segmentation of the ERP map series [22] were used. The landscape of each map, i.e., the spatial configuration of its potential distribution, was assessed numerically by the locations of the centroids of the positive and negative map areas [21].

2.7 Statistical Analysis

Statistical analysis was performed on the GFP area of 21 electrodes sites within the time range of difference waveform of cross-modal interaction (100-250 ms).

Two groups consisted of Chinese and Korean separately and two conditions were meaningful and meaningless. Five sites were prefrontal line, frontal line, central line, parietal line, and occipital line, respectively. ERP was analyzed with three-way ANOVAs with a repeated measure (group x condition x electrode site). Four electrodes sites such as prefrontal line (FP1, FPz, Fp2), frontal line (F7, F3, Fz, F4, F8), central line (T7, C3, Cz, C4, T8), parietal midline (P7, P3, Pz, P4, P8), and occipital line (O1, Oz, O2) sites were used.

3 Results

3.1 Electrophysiological Data

A three-way repeated measures ANOVA conducted on the mean amplitude yielded main effects of group ($F_{1,26} = 7.31, p = 0.01$) and condition ($F_{1,26} = 9.63, p < 0.01$), and an interaction effect between group and condition ($F_{1,26} = 5.40, p = 0.04$). No main effect of electrode location was found ($F_{1,26} = 1.44, p = 0.282$). Korean subjects showed a larger mean amplitude relative to Chinese subjects for the both condition ($F_{1,26} = 9.73, p < 0.01$) (Fig. 2).

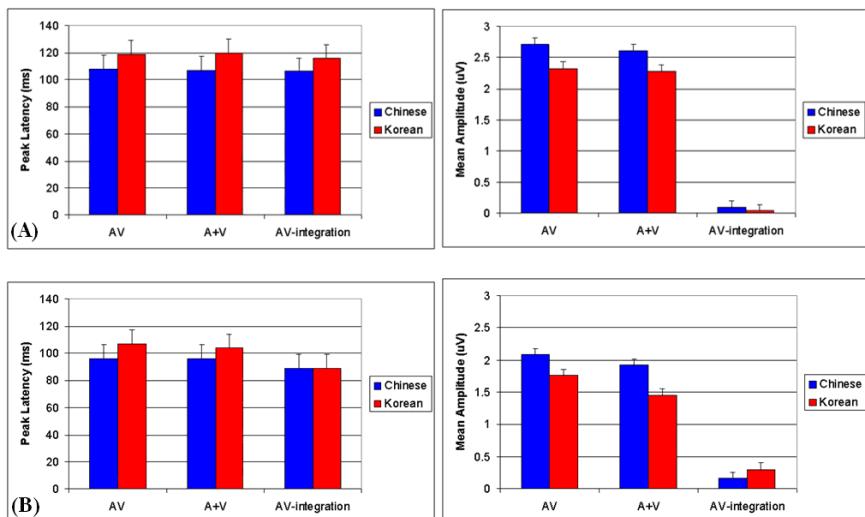


Fig. 2. Mean peak latency (left) and amplitude (right) values are displayed for the two subjects groups in the meaningful (A) and meaningless (B) stimuli perception

The mean amplitude response for the Korean group was significantly less in meaningful stimuli condition than in the meaningless stimuli condition ($F_{1,26} = 11.41, p < 0.01$). There was no significant difference between the two conditions ($F_{1,26} = 0.74, p = 0.58$) in Chinese group. Irrespective of group, the peak later for the meaningful stimuli (Chinese = 106 ms; Korean = 116 ms) relative to the

meaningless stimuli (Chinese = 89 ms; Korean = 89 ms). A repeated measures three-way ANOVA [group x condition x electrode location] conducted on the peak latency measure yielded a significant main effect of condition ($F_{1,26} = 16.40, p < 0.01$), indicating that the peak of the integration occurred later in time for the meaningful stimuli relative to the meaningless stimuli. No other main effects or interaction effects reached significance.

3.2 Event-Related Potential (ERP) Map Sequences

Global Map Dissimilarity was computed between all maps of the mean map sequence of the Korean (k) and Chinese (c) (Fig. 3A and 3B, right), and between all maps of the mean map sequence of the Chinese with themselves (Fig. 3A and 3B, left), resulting in what is called now “landscape correlation matrices”.

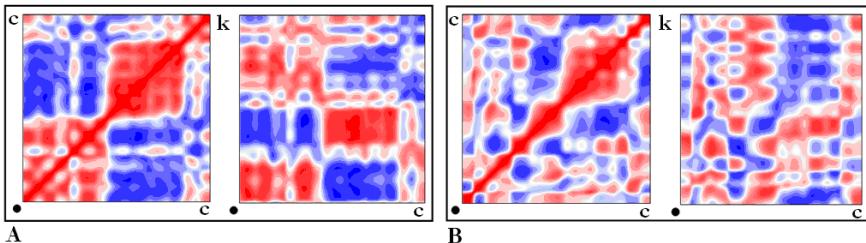


Fig. 3. Matrices of correlations between map landscapes of meaningful (A) and meaningless (B) stimuli. Original is the left lower corner (dot); time runs up and to the right; thus, maps of equal latency are compared along the diagonal from the lower left corner to the upper right corner. Left graph: mean ERP maps of the Chinese vs themselves; Right graph: mean ERP maps of the Chinese (horizontal) vs. those of the Korean (vertical). The color scale ranges from light red (high positive correlation) to dark red (low positive correlation) to dark blue (low negative correlation) to light blue (high negative correlation). Note the strong asymmetry of the correlations of Chinese vs. Korean data in the meaningful stimuli, where the Chinese's (horizontal) AV integration microstate starts much earlier than that of the Korean (vertical).

3.3 N1 Map Landscapes

The mean locations of the landscape centroids of the N1 maps at group centre latency for meaningful and meaningless stimuli in Chinese and Korean is shown in Fig. 4. None of the landscape differences as represented by the centroids reached significant double-ended p-values in meaningful stimuli. In meaningless stimuli, there were statistical differences of interest along the anterior-posterior axis: the positive (posterior) centroid was more anterior ($P < 0.03$) and the negative (anterior) centroid was more posterior ($P < 0.07$) in Korean than in Chinese. Thus, the anterior-posterior distance between centroids was smaller in Korean than in Chinese.

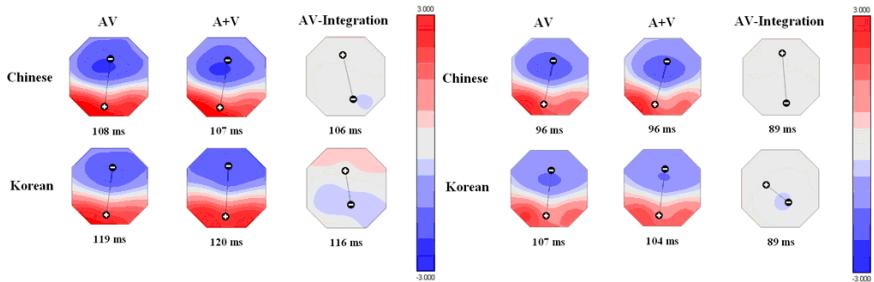


Fig. 4. The locations of the mean (across subjects) map centroids at the peak latencies for the meaningful (left) and the meaningless (right) stimuli per experimental conditions. Anterior centroids are negative (-), posterior centroids are positive (+). The negative centroid of the AV integration was more posterior in Chinese relative to Korean in meaningful stimuli perception whereas the posterior centroid more anterior in the Chinese than in the Korean in meaningless stimuli.

4 Discussion

The present study was able to find evidence of both non-phonetic and phonetic audio-visual interactions in the ERPs to the same AV stimuli. The differences in the sum of the ERPs to the unimodal A and V stimuli and in ERPs to AV stimuli indicated interactions presumably based on temporal of the A and V components of the AV stimuli. These interactions appeared to be similar for both Chinese and Korean groups. In addition, the differences in the ERPs to the meaningful and meaningless of AV stimuli probably reflect multisensory interactions in phonetic processing. When acoustic and visual phonemes were meaningful, they formed a natural multisensory interaction stimulus. The results of the reaction-time experiment are clearly shown. When subjects categorized acoustic pitch contours, incongruent visual pitch prolonged reaction times ($P < 0.01$) and congruent visual pitch ($P < 0.05$) shortened the reaction times. When subjects categorized visual pitch, incongruent acoustic pitch prolonged reaction times ($P < 0.01$), but the congruent ones did not affect reaction times. These results demonstrate that phonetic-level interactions between acoustic and visual affect identification speed of acoustic/visual perception. The phonetically incongruent audio-visual pitches are recognized more slowly than the congruent ones. The comparison of ERPs to the Chinese and Korean in audio-visual pitch contours perception revealed significant differences at latencies of 106 and 116 ms after the sound onset. Both interactions could be explained by modulated activity in parieto-occipital and temporal regions. Importantly, the non-phonetic interactions, which were probably generated in the sensory-specific regions, were earlier than the first phonetic interaction. All phonetic interactions were probably generated in the high-level multisensory regions. The results suggest that sensory-specific and multisensory cortices are involved in audio-visual processing at separate latencies and that they are sensitive to different features of audio-visual stimuli. The present results corroborate previous findings showing that visual stimuli have access to the

early levels of auditory processing hierarchy [23-28] and support the auditory integration models. Integration of auditory and visual non-speech information is primarily based on temporal and spatial coincidence of the stimuli. These mechanisms are important in audio-visual integration of speech as well. However, seeing and hearing speech provide also phonetic information. Therefore, both general and speech-specific multisensory mechanisms might be important in audio-visual perception of speech [23,27]. Some comments on the rationale for analyzing the sources of the AV - (A+V) difference wave may be discussed. This difference wave may have a non-zero value whenever the bimodal AV stimulus elicits a different amount or a different configuration of neural activity from the linear sum of the activities elicited by the unimodal A and V stimuli. This differential neural activity could result from an enhancement (or suppression) of activity within one (or both) of the unimodal source configuration, or it could arise from a different source not activated by either unimodal stimulus alone. In any case, this differential neural activity due to cross-modal interaction is, in principle, localizable to a particular brain region or regions. Previous studies have made inferences about the localization of such interactive neural events on the basis of their associated scalp fields manifest in the AV - (A+V) difference wave [29,30]. In the present study, scalp topographical evidence was supplemented by map landscape in order to estimate the locations of the contributing sources.

5 Conclusions

The current study demonstrates that the audiovisual interaction is an indicator for investigating the automatic processing of suprasegmental information in tonal language. The use of multiple language groups is important for showing language-related differences in the relative importance of perceptual dimensions that may influence the magnitude of the response to pitch contours. The convergence of ERP and earlier behavioral evidence further supports the notion that the relative saliency of perceptual dimensions underlying lexical tones may influence the automatically processing of pitch contours at early stages of speech perception. This finding gives support for the view that both sensory-specific and heteromodal cortices are involved in the AV integration process at separate latencies and are sensitive to different features of AV stimuli.

Acknowledgments. This study was supported by the KFAS International Scholar Exchange Fellowship Program, 2006-2007 and a grant (M103KV010007 04K2201) from Brain Research Center of the 21st Century Frontier Research Program funded by the Ministry of Science and Technology of Republic of Korea.

References

1. Tervaniemi, M., Rytkönen, M., Schröger, E., Ilmoniemi, R., Näätänen, R.: Superior formation of cortical memory traces for melodic patterns in musicians. *Learning & Memory* 8, 295-300 (2001)

2. Schröger, E., Tervaniemi, M., Wolff, C., Näätänen, R.: Preattentive periodicity detection in auditory patterns as governed by time and intensity information. *Brain Res. Cogn. Brain Res.* 4, 145–148 (1996)
3. Alain, C., Woods, D.L., Ogawa, K.H.: Brain indices of automatic pattern processing. *NeuroReport* 6, 140–144 (1994)
4. Alho, K., Tervaniemi, M., Huotilainen, M., Lavikainen, J., Tiitinen, H., Ilmoniemi, R.J., Knuutila, J., Näätänen, R.: Processing of complex sounds in the human auditory cortex as revealed by magnetic brain responses. *Psychophysiology* 33, 396–375 (1996)
5. Scherg, M., Vajsar, J., Picton, T.W.: A source analysis of the late human auditory evoked potentials. *J. Cogn. Neurosci.* 5, 363–370 (1989)
6. Näätänen, R.: Attention and Brain Function. Erlbaum, Hillsdale (1992)
7. Nyman, G., Alho, K., Iaurinen, P., Paavilainen, P., Radil, T., Rainikainen, K.: Mismatch negativity (MMN) for sequences of auditory and visual stimuli: evidence for a mechanism specific to the auditory modality. *Electroenceph. Clin. Neurophys.* 77, 436–444 (1990)
8. Brattico, E., Winkler, I., Näätänen, R., Paavilainen, P., Tervaniemi, M.: Simultaneous storage of two complex temporal sound patterns in auditory sensory memory. *NeuroReport* 13, 1747–1751 (2002)
9. Ritter, W., Deacon, D., Gomes, H., Javitt, D.C., Vaughan Jr., H.G.: The mismatch negativity of event-related potentials as a probe of transient auditory memory: a review. *Ear Hear* 16, 52–67 (1995)
10. Tales, A., Newton, P., Troscianko, T., Butler, S.: Mismatch negativity in the visual modality. *NeuroReport* 10, 3363–3367 (1999)
11. Näätänen, R.: The role attention in auditory information processing as revealed by event-related potentials and other brain measures of cognitive function. *Behav. Brain Sci.* 13, 201–288 (1990)
12. Czigler, I., Balázs, L., Pató, L.G.: Visual change detection: event-related potentials are dependent on stimulus location in humans. *Neurosci. Lett.* 364(3), 149–153 (2004)
13. Yucel, G., Pettv, C., McCarthv, G., Belger, A.: Graded visual attention modulates brain response evoked by task-irrelevant auditory pitch changes. *J. Cogn. Neurosci.* 17(2), 1819–1828 (2005)
14. Czigler, I., Balázs, L., Winkler, I.: Memory-based detection of task-irrelevant visual changes. *Psychophysiology* 39, 869–873 (2002)
15. Heslenfeld, D.J.: Visual mismatch negativity. In: Polish, J. (ed.) *Detection of change: event-related potential and fMRI findings*, pp. 41–60. Kluwer Academic Publishers, Dordrecht (2003)
16. Cammann, R.: Is there a mismatch negativity (MMN) in the visual modality? *Behv. Brain Sci.* 13, 234–235 (1999)
17. Pazo-Alvarez, P., Cadaveira, F., Amenedo, E.: MMN in the visual modality: a review. *Biol. Psychology* 63, 1999–1236 (2003)
18. Woods, D.J., Alho, K., Algazi, A.: Intermodal selective attention. I. Effects on event-related potentials to lateralized auditory and visual stimuli. *Electroenceph. Clin. Neurophys.* 82, 341–355 (1992)
19. Alho, K., Woods, D.L., Algazi, A., Näätänen, R.: Intermodal selective attention. II. Effects of attentional load on processing of auditory and visual stimuli in central space. *Electroenceph. Clin. Neurophys.* 82, 356–368 (1992)
20. Smid, H.G.O.M., Jakob, A., Heinze, H.J.: An event-related brain potential study of visual selective attention to conjunctions of color and shape. *Psychophysiology* 36, 264–279 (1999)

21. Wackermann, J., Lehmann, D., Michel, C.M., Strik, W.K.: Adaptive segmentation of spontaneous EEG map series into spatially defined microstates. *Int. J. Psychophys.* 14, 269–283 (1993)
22. Lehmann, D., Skrandies, W.: Segmentation of evoked potentials based on spatial field configuration in multichannel recordings. In: McCallum, W.C., Zappoli, R., Denoth, F. (eds.) *Cerebral Psychophysiology: Studies in Event-Related Potentials. Electroencephalography and Clinical Neurophysiology* (suppl. 38), pp. 506–512 (1986)
23. Klucharev, V., Möttönen, R., Sams, M.: Electrophysiological indicators of phonetic and non-phonetic multisensory interactions during audiovisual speech perception. *Brain Res. Cog. Brain Res.* 18, 65–75 (2003)
24. Möttönen, R., Schurmann, M., Sams, M.: Time course of multisensory interactions during audiovisual speech perception in humans: a magnetoencephalographic study. *Neurosci. Let.* 363, 112–115 (2004)
25. Sams, M., Aulanko, R., Häimäläinen, M., Haru, R., Lounasmaa, O.V., Lu, S.-T.: Seeing speech: visual information from lip movements modifies activity in the human auditory cortex. *Neurosci. Let.* 127, 141–145 (1991)
26. Besle, J., Fort, A., Delpuech, C., Giard, M.H.: Bimodal speech: early suppressive visual effects in human auditory cortex. *Euro. J. Neurosci.* 20, 2225–2234 (2004)
27. Calvert, G.A., Thesen, T.: Multisensory integration: methodological approaches and emerging principles in the human brain. *J. Phys. Paris* 98, 191–205 (2004)
28. van Wassenhove, V., Grant, K.W., Poeppel, D.: Visual speech speeds up the neural processing of auditory speech. *Proc. Natl. Acad. Sci. USA* 102, 1181–1186 (2005)
29. Foxe, J.J., Morocz, I.A., Murray, B.A., Higgins, B.A., Javitt, D.C., Schroeder, C.E.: Multisensory auditory-somatosensory interactions in early cortical processing revealed by high-density electrical mapping. *Cog. Brain Res.* 10, 77–83 (2000)
30. Giard, M.H., Peronnet, F.: Audio-visual integration during multimodal object recognition in humans: a behavioral and electrophysiological study. *J. Cog. Neurosci.* 1, 473–490 (1999)

A Model for Neuronal Signal Representation by Stimulus-Dependent Receptive Fields

José R.A. Torreão, João L. Fernandes, and Silvia M.C. Victer

Instituto de Computação, Universidade Federal Fluminense
24210-240 Niterói RJ, Brazil

Abstract. Image coding by the mammalian visual cortex has been modeled through linear combinations of receptive-field-like functions. The spatial receptive field of a visual neuron is typically assumed to be signal-independent, a view that has been challenged by recent neurophysiological findings. Motivated by these, we here propose a model for conjoint space-frequency image coding based on stimulus-dependent receptive-field-like functions. For any given frequency, the parameters of the coding functions are obtained from the Fourier transform of the stimulus. The representation is initially presented in terms of Gabor functions, but can be extended to more general forms, and we find that the resulting coding functions show properties that are consistent with those of the receptive fields of simple cortical cells of the macaque.

1 Introduction

The receptive field is a fundamental concept for the description of neuronal activity. In the classical picture, the receptive field of a visual neuron becomes defined by the region of space where light stimuli evoke neuronal responses, and by the nature of such responses [1]. For instance, in the so-called ON regions of the receptive field, an increment in luminance evokes an increased neuronal response, the inverse being true of an OFF region. In this classical description, the spatial organization of the receptive field is assumed fixed, and the neuronal response to a time-invariant stimulus is obtained as the result of filtering the stimulus by the fixed receptive field. On the other hand, linear combinations of receptive-field-like functions (whose most common theoretical model is Gabor's elementary signals: gaussian modulated complex exponentials [2]) have been used for describing conjoint space-frequency signal coding by the mammalian visual cortex [3].

Lately, the classical view of a signal-independent receptive field has been challenged by neurophysiological findings which indicate that the spatial receptive-field structure changes with the neuronal input [4,5,6]. Motivated by such findings, we here propose a conjoint space-frequency representational model in terms of signal-dependent coding functions. More specifically, we show that any given signal can be accurately represented by a set of receptive-field-like functions - well localized both in space and in frequency - whose defining parameters are obtained from the signal's Fourier transform.

2 Deriving the Representation

For simplicity, we first derive the proposed representation for one-dimensional signals, and later extend our results to the two-dimensional case. We also start by considering Gabor functions as the receptive-field model, but later show how this can be generalized.

Let $I(x)$ be a square-integrable signal defined over an unbounded domain, such that its Fourier transform exists. The following identity holds:

$$I(x) = \int_{-\infty}^{\infty} d\omega e^{i\omega x} * e^{i\omega(x-a)} e^{-\frac{(x-a)^2}{2\sigma^2}} \quad (1)$$

where the asterisk denotes a convolution, and where a and σ are functions of the frequency ω , given by

$$\sigma(\omega) = \frac{1}{(2\pi)^{3/2}} |\tilde{I}(\omega)| \quad (2)$$

and

$$a(\omega) = -\frac{\varphi_{\tilde{I}}(\omega)}{\omega}, \quad \omega \neq 0 \quad (3)$$

with $|\tilde{I}|$ and $\varphi_{\tilde{I}}$ denoting, respectively, the amplitude and the phase of the signal's Fourier transform, $\tilde{I}(\omega)$. When $\omega = 0$, a is defined as zero.

Eq. (1), with a and σ as above, can be easily proven equivalent to the inverse Fourier transform, as shown in the Appendix. It still expresses the signal through an expansion in a complex-exponential basis set, but the ‘coefficients’ of such expansion are functions both of position and of frequency, given by

$$e^{i\omega(x-a)} e^{-\frac{(x-a)^2}{2\sigma^2}} \quad (4)$$

with σ and a as in equations (2) and (3). Thus, we have obtained a conjoint space-frequency representation in terms of Gabor functions whose parameters are related to the signal's Fourier transform. Making the convolution operation explicit in Eq. (1), it can be formally rewritten as (see Appendix)

$$I(x) = \int_{-\infty}^{\infty} d\omega \phi_{\omega}(x) [\psi_{\omega}(x) \cdot \phi_{\omega}(x)] \quad (5)$$

where $\phi_{\omega}(x)$ denotes the complex exponential basis function, $\psi_{\omega}(x)$ is the Gabor coding function, and the term under the square brackets is their inner product. Comparing Eq. (5) to the standard form for a signal expansion on a basis set,

$$I(x) = \int_{-\infty}^{\infty} d\omega \phi_{\omega}(x) [I(x) \cdot \phi_{\omega}(x)] \quad (6)$$

we see that, in so far as the frequency ω is concerned, the coding function $\psi_{\omega}(x)$ is equivalent to the signal $I(x)$. Thus, the set $\{\psi_{\omega}(x)\}$ of stimulus-dependent

receptive-field-like functions provides a complete space-frequency representation of $I(x)$.

The proposed representation can be easily extended to two-dimensional signals, in which case we find

$$I(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} d\omega_x d\omega_y e^{i(\omega_x x + \omega_y y)} * e^{i[\omega_x(x-a) + \omega_y(y-a)]} e^{-\left[\frac{(x-a)^2 + (y-a)^2}{2\sigma^2}\right]} \quad (7)$$

where

$$\sigma(\omega_x, \omega_y) = \frac{1}{(2\pi)^{3/2}} \sqrt{|\tilde{I}(\omega_x, \omega_y)|} \quad (8)$$

and

$$a(\omega_x, \omega_y) = -\frac{\varphi_{\tilde{I}}(\omega_x, \omega_y)}{\omega_x + \omega_y}, \quad (\omega_x, \omega_y) \neq (0, 0) \quad (9)$$

with a being defined as zero, when $(\omega_x, \omega_y) = (0, 0)$. The coding functions now take the form

$$e^{i[\omega_x(x-a) + \omega_y(y-a)]} e^{-\left[\frac{(x-a)^2 + (y-a)^2}{2\sigma^2}\right]} \quad (10)$$

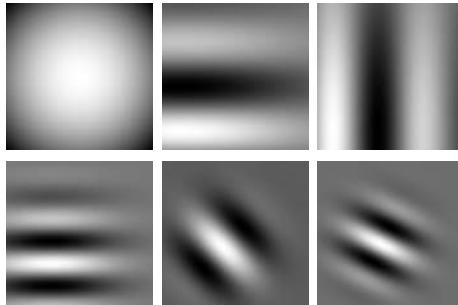


Fig. 1. Examples of the coding functions of Eq. (10), obtained from a 16×16 fragment of the original image in Fig. 3. From top down and left to right, the represented frequencies, (ω_x, ω_y) , are $(0,0)$, $(0,1)$, $(1,0)$, $(0,2)$, $(1,1)$ and $(1,2)$.

It should be remarked that, although based on Gabor functions, the representation in Eq. (7) is quite distinct from a Gabor-transform expansion [7]. The latter is a linear expansion on a set of Gabor functions whose parameters must be chosen a priori, the coefficients of such expansion being obtained from a Gabor transform, which corresponds to a gaussian-windowed Fourier transform. In our case, the expansion ‘coefficients’ are Gabor functions whose parameters are obtained from a standard Fourier transform.

Examples of such functions (computed from a 16×16 fragment of the original image in Fig. 3), appear in Fig. 1. Since the Fourier spectrum of typical images is



Fig. 2. Top: original images. Bottom: Conjoint space-frequency representation computed using 3×3 windows, for the frequencies $(0,0)$, $(0,1)$, $(1,0)$, $(0,2)$, $(1,1)$ and $(1,2)$.

concentrated at the lower frequencies, these functions tend to become narrower - and thus their bandwidth tend to increase - with increasing frequency. This kind of behavior has been observed for the bandwidth of simple cortical cells of the macaque [8], and, in a more recent work [9], receptive fields which are low-pass and broadly tuned in orientation, similarly as the DC coding function of Fig. 1, have also been found in the macaque's visual cortex. Such receptive fields are not predicted by the usual models of cortical coding [39].

Eq. (7) is an exact mathematical result which holds for any square-integrable signal defined over an infinite domain, but it remains approximately valid over finite windows, with different σ - and a -values computed at each window. Fig. 2 shows the representation obtained for a set of natural images [10], considering only six low frequency components computed over 3×3 windows. The average representation error, over the set, is 3.9%. In Fig. 3, we present an image representation computed only from the DC component of Eq. (7). Although the error is higher (6.6%), the representation is visually quite faithful to the original input.

3 Generalizing the Representation

The proposed conjoint space-frequency representation can be generalized in several ways. For instance, we can use rotated complex exponentials as basis functions (the integrals are from $-\infty$ to ∞):

$$I(x, y) = \int \int d\omega_x d\omega_y e^{i(\omega_x x_\theta + \omega_y y_\theta)} * e^{i[\omega_x(x-a)_\theta + \omega_y(y-a)_\theta]} e^{-\left[\frac{(x-a)^2 + (y-a)^2}{2\sigma^2}\right]} \quad (11)$$

where

$$\begin{cases} x_\theta = x \cos \theta + y \sin \theta \\ y_\theta = -x \sin \theta + y \cos \theta \end{cases} \quad (12)$$

in which case a and σ will be given, respectively, in terms of the Fourier transforms of the rotated and the inversely rotated signals, θ being the rotation angle.



Fig. 3. Left: Original image. Right: Approximation obtained through the DC component of Eq. (7), computed over 3×3 windows. Representation error of 6.6%.

We can also generalize Eq. (7) to Gabor basis functions, as

$$I(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \int \int d\omega_x d\omega_y \int \int d\alpha d\beta e^{i(\omega_x x + \omega_y y)} e^{-\left[\frac{(x-\alpha)^2}{2\sigma_x^2} + \frac{(y-\beta)^2}{2\sigma_y^2}\right]} * e^{i[\omega_x(x-a) + \omega_y(y-a)]} e^{-\left[\frac{(x-a)^2 + (y-a)^2}{2\sigma^2}\right]} \quad (13)$$

where σ_x and σ_y are constants, while σ and a remain as in the original model.

Similarly to the original representation, the variants of Eqs. (11) and (13) can be easily proven equivalent to the inverse Fourier transform, Eq. (7) being the particular case of (11) for $\theta = 0$, and the limiting case of (13) when both σ_x and σ_y tend to zero. Eq. (13) can also be easily extended to the case of rotated Gabor basis functions, and, in this more general guise, it will be comparable to the conjoint space-frequency models of cortical image representation [30], which express a given image in terms of spatially localized, oriented and bandpass basis functions.

The proposed conjoint space-frequency representation is not restricted to Gabor functions, the adequacy of which, as a model for receptive fields, has been questioned by some [1]. Being $F(x, y; S)$ a generic, unnormalized 2D function, depending on the spatial parameter S , we can generalize Eq. (7) as

$$I(x, y) = \int \int d\omega_x d\omega_y e^{i(\omega_x x + \omega_y y)} * e^{i[\omega_x(x-a) + \omega_y(y-a)]} F(x - a, y - a; S) \quad (14)$$

where a remains defined by Eq. (9), and where $S(\omega_x, \omega_y)$ is to be found from the relation

$$|\tilde{F}(0, 0; S(\omega_x, \omega_y))| = \frac{1}{(2\pi)^2} |\tilde{I}(\omega_x, \omega_y)| \quad (15)$$

Eq. (14), which can be easily proven following the steps in the Appendix, still employs a complex-exponential basis, but along with a more general coding function. The extension to the case where the basis and the coding functions share this more general form is straightforward; so is the derivation of the variant with rotated versions of these functions. We have tested the case where F is the physical point-spread function of an imaging system [2], and obtained representations only slight inferior to those yielded by the Gabor functions. Further experimentation with alternative coding functions is still needed.

4 Conclusion

We have presented a new conjoint space-frequency representational model, based on receptive-field-like coding functions whose parameters are obtained from the Fourier transform of the input signal. For a typical image, the coding functions show properties that are consistent with those of the receptive fields of cortical cells, what indicates that they could prove useful for neurophysiological modeling (the fact that the coding functions depend on the Fourier transform of the signal does not hinder the model's neurophysiological plausibility, since the representation is not substantially affected if a transform under a broad gaussian window is considered). Our representation might also be advantageous in other domains, such as compressive coding. For instance, sharp edges usually present a problem for efficient compression, since these are localized spatial features, and thus spread out in frequency. In terms of the proposed model, an edge would lead to a number of coding functions that are broad in space, and thus localized in frequency. Compressive coding of these functions, instead of the original signal, might yield better compaction.

Acknowledgment. This work has been partially supported by CNPq-Brasil.

References

1. Hubel, D.H., Wiesel, T.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* 160, 106–154 (1962)
2. Marcelja, S.: Mathematical description of the responses of simple cortical cells. *J. Opt. Soc. Am. A* 70, 1297–1300 (1980)
3. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381, 607–609 (1996)
4. Allman, J., Miezin, F., McGuinness, E.: Stimulus specific responses from beyond the classical receptive field: Neurophysiological mechanisms for local-global comparison in visual neurons. *Annu. Rev. Neuroscience* 8, 407–430 (1985)
5. David, S.V., Vinje, W.E., Gallant, J.L.: Natural stimulus statistics alter the receptive field structure of V1 neurons. *The Journal of Neuroscience* 24(31), 6991–7006 (2004)
6. Bair, W.: Visual receptive field organization. *Current Opinion in Neurobiology* 15(4), 459–464 (2005)
7. Daugman, J.: Complete discrete 2-D Gabor transform by neural networks for image analysis and compression. *IEEE Trans. on Acoustics, Speech, Signal Processing* 36(7), 1169–1179 (1988)
8. DeValois, R.L., Albrecht, D.G., Thorell, L.G.: Spatial frequency selectivity of cells in macaque visual cortex. *Vision Res.* 22, 545–559 (1987)
9. Ringach, D.L.: Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *J. Neurophysiol.* 88, 455–463 (2002)
10. van Hateren, J.H., van der Schaaf, A.: Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc. R. Soc. Lond. B* 265, 359–366 (1998)
11. Stork, D.G., Wilson, H.R.: Do Gabor functions provide appropriate descriptions of visual cortical receptive fields? *J. Opt. Soc. Am. A* 7, 1362–1373 (1990)
12. Hecht, E.: Optics. Addison-Wesley, Reading (1990)

Appendix

Here we prove the identity in Eq. (1), by taking the Fourier transform of both sides of the equation. Changing the integration variable to ω' , we have

$$\tilde{I}(\omega) \equiv \mathcal{F}\{I(x)\} = \int_{-\infty}^{\infty} d\omega' \mathcal{F}\{e^{i\omega' x} * e^{i\omega'(x-a)} e^{-\frac{(x-a)^2}{2\sigma^2}}\} \quad (16)$$

and using the convolution property of the transform,

$$\tilde{I}(\omega) = \int_{-\infty}^{\infty} d\omega' \mathcal{F}\{e^{i\omega' x}\} \times \mathcal{F}\{e^{i\omega'(x-a)} e^{-\frac{(x-a)^2}{2\sigma^2}}\} \quad (17)$$

The two transforms under the integral are well-known, and we obtain

$$\tilde{I}(\omega) = (2\pi)^{3/2} \int_{-\infty}^{\infty} d\omega' \delta(\omega - \omega') \sigma(\omega') e^{-i\omega a} e^{-\frac{\sigma^2(\omega')}{2}(\omega - \omega')^2} \quad (18)$$

where δ is Dirac's delta function. Using the sifting property of the delta, we finally arrive at

$$\tilde{I}(\omega) = (2\pi)^{3/2} \sigma(\omega) e^{-i\omega a} \quad (19)$$

from which the relations in Eqs. (2) and (3) follow.

Also note that, making the convolution in the integrand of Eq. (1) explicit, it can be rewritten as

$$I(x) = \int_{-\infty}^{\infty} d\omega e^{i\omega x} \int_{-\infty}^{\infty} d\xi e^{-i\omega\xi} e^{i\omega(\xi-a)} e^{-\frac{(\xi-a)^2}{2\sigma^2}} \quad (20)$$

where the second integral is the inner product of the exponential basis function and the Gabor coding function, what proves Eq. (5).

Area Chip Consumption by a Novel Digital CNN Architecture for Pattern Recognition

Emil Raschman and Daniela Ďuračková

Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology,
Bratislava, Slovakia
{emil.raschman,daniela.durackova}@stuba.sk

Abstract. The implementation for the digital neural networks on a chip requires a lot of chip area consumption. Our contribution paper deals therefore with the design of a novel type of digital CNN architecture focused on pattern recognition application. The novel designed network we compare with another two CNN implementation of digital network on a chip used for pattern recognition by the selected parameters as the speed and chip area consumption. From the comparison we can recognize that our proposed digital CNN network is the best from the other ones.

Keywords: CNN, pattern recognition, digital implementation of CNN networks.

1 Introduction

Between the quickest ways of pattern recognition belong neural networks implementation. Their advantage is gigantic computing power due to parallel data processing and disadvantage is circuit's trickiness due to a lot of computational elements which provide parallel processing. Advantage of some neural networks is their ability to learn, gratitude to which we needn't to develop composite software for data processing that are occupying a lot of place in memory. System for data processing is creating simple learning with special training aggregate.

For image and pattern processing most frequently we use so-called cellular neural networks marked as CNN. This type of networks contains a lot of cells (computing elements), which are interconnected analogically as the neurons in humane brain. An analogue architecture of this network is proposed and presented in 1988 L. O. Chua aL. Yang [12]. Basic principle of the CNN contains literatures [3,4,5,6].

2 Implementation of CNN on a Chip

The CNN network is possible to design as an analog or digital circuits. Various hardware implementations are present in the literatures [7,8,9,10,11,12]. The advantage of analog implementation is smaller chip area and advantage of digital

implementation is easy modification for weight coefficients by the process of learning. We are in our work focused to digital neural network, which advantage is relatively easy implementation in the FPGA circuits. Basic property of this type of the neural network are chip area consumption, high speed and low consumption.

In the present time VLSI circuits and present technology achieve the manufacture of the chip with very high density of integration, what make possible hardware implementation with high number of neurons (neural cells) on the chip. Even though henceforward stays the biggest problem for hardware realization of neural networks from the point of view on chip area consumption and from depended price. So than they could contain the cells occupying minimal area on chip. In that digital design the most area is occupied by multiplier and adder, hence we need the proposed alternative way of multiplication and added this in consideration of consumption chip area.

The principle of the CNN networks is based on very simple principle similar to those of biological neuron. Network is consisting of quantity basic computing elements so-called cells. Incoming inputs of cell are multiplying correspondent weight coefficient, adding and then making conversion through the transfer function. Because all cells realize information's processing in parallel, calculation power CNN network is direct proportional to the number of cells. The more cells contains the network, the more information achieve synchronized processing. Therefore in design of the CNN network effort is focused to minimize cell size and thereby provide for maximum number of cells on chip. Size of the chip is one of biggest problems at designing CNN network. The most area of chip takes hardware multiplier unit, therefore we are searching other alternative multiplication.

2.1 The Multiplication of the Signal Using the AND Gate

How to alternate the possibility of multiplication is to design circuit, which multiplication input values and weight coefficient are by means of AND gate. The method of multiplication is based on the fact, that by multiplication the input value must be converted to the time interval from begin of time-window and the weight value has to be special picked, so by the timing starts multiplication. We proposed a special coding for the weights. The value of weight signal must be distributed symmetric in time-window.

We used a special system of 15 parts, e.g. one cycle is divided into 15 parts. In such a time period is possible to code 16 various values weight or inputs. Decomposition signal for corresponding weight values are displayed in Fig. 2.

As the example we can input value $x = 8/15$ multiply with the weight $w = 5/15$, according the Fig. 2 and input values must be transformed to the time interval from begin time axis corresponding with input size. We can get the output signal $y = 3/15$ from the time interval. The real value from the $x \cdot w = 0.18$ and the result after Fig. 2 is $3/15 = 0.2$.

Natural property of this proposed method of multiplication is rounding. For verification effect of the rounding on the result CNN network we are create

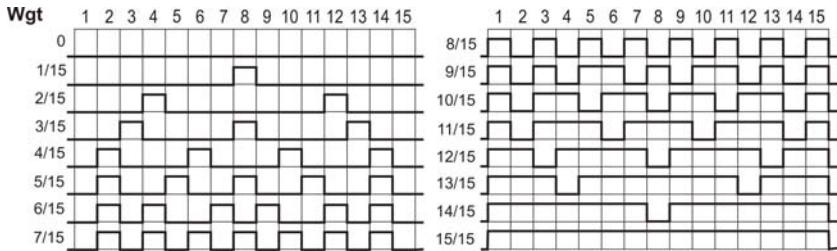


Fig. 1. Weights in the proposed 15s system of the timing

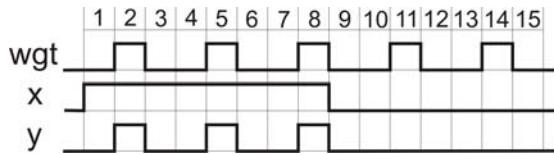


Fig. 2. An example of the evaluation for weight $wgt=5/15$ and the input $x=8/15$

simulator as macro at Visual Basic for Application in MicrosoftExcel. We used the simulator to recognize, that we can this rounding neglect. For example existence of multiple rounding the intermediate result caused that the final result network will be delayed about one iteration later, than in example without rounding.

The value of the error of computing depends from factors conjunction. A decomposition error is in the Fig. 3. On this picture a) is decomposition absolute error. They maximum value is 0,031. On the picture b) is displayed absolute error in percentage. On the figure we can see effect of the rounding, where for numbers, that were rounding to zero is absolute error 100%, but those number

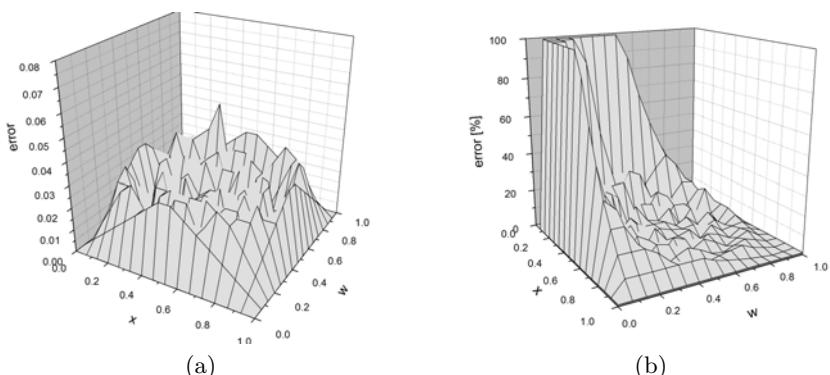


Fig. 3. Decomposition errors depended from factor conjunction

are so small, that we can ignore them. Proper by selection weight coefficients is possible shift calculation in the areas with minimum errors.

2.2 The Cell of the Proposed CNN

The proposed circuit is a digital synchronous circuit. Designed circuit has been realized by means of descriptive language VHDL in development environment Xilinx. The cell contains several sub-circuits. The block diagram of cell is in the Fig. 4. On the inputs of cell are 9 weight and 9 input signals and their corresponding signs. Eight input signals are attached on the output of neighboring cells and one input, fifth in sequence is attached on own output of cell, because in the CNN theory is the cell neighbor to itself. Then the inputs are multiplied with weights in logical *AND* gate and sign inputs are compared with the logical *XOR* gate. Size of the weight must be specially decomposed in time according to Fig. 11. This will ensure multiplication of the signals at the *AND* gate. If the result of multiplication is positive (output of *XOR* gate is log 0), then the result is added to counter and if it is negative (output of *XOR* gate is log 1), then the result is subtracted from counter. Output signal of counter is transformed in block *transfer function*, which realized threshold logic function.

Converted signal is coming through multiplexer *mx* to the block *converter*. Multiplexer *mx* allows enter input values to the network. Block *converter* has two functions: it contains register, where is saved the result, from it is possible to read (*data out*) and circuit converting results on time interval corresponding size of results (*statex_o*, *sign_o*), which feed into surrounding cells.

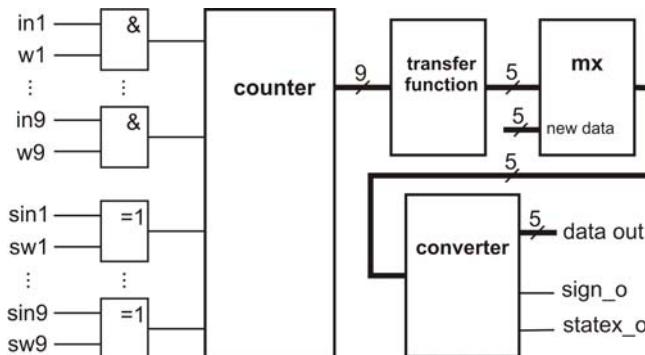


Fig. 4. Block diagram cell of the CNN

2.3 A Novel Architecture of the CNN Network

CNN network consists from a field of cells, in that every cell is coupled with everyone of the nearest neighbor i.e. output one's cell is the input to all surrounding cells. We can see, that every cell is on the fifth input coupled with its own output, because in the CNN theory is every cell neighbor to itself too.

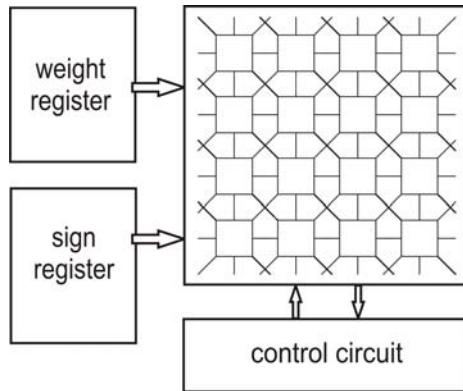


Fig. 5. Block diagram of CNN network

General design CNN network is in the Fig. 5. Complete network contain network of coupled cells, weight and sign register where from weights are feed individual cell and circuit for synchronization and control network.

3 The Comparison of Our Network with Others

The new architectures CNN network we compared with other digital CNN network. For comparison we selected following networks:

- Our previous CNN network [13].
- Standard digital CNN network with parallel signed multipliers.

All network for comparison were designed for processing 5-bit data, where 4 bits represented values and highest bit represented sign of values.

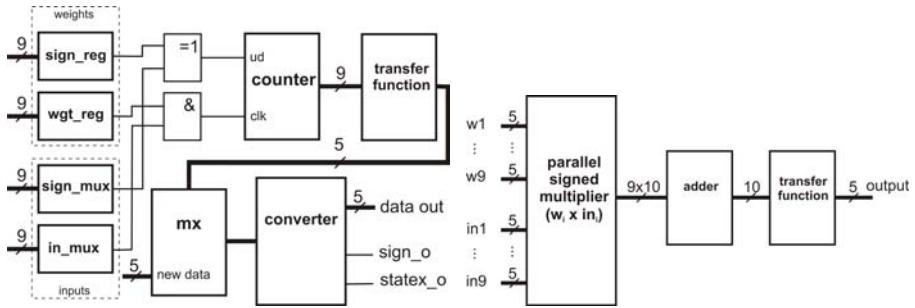
3.1 Our Previous CNN Network

This network is previous to our new CNN architecture. Network was proposed as the synchronous digital circuit, which also is based on utilization of multiplication signals distribution in the time by means of logical *AND* gate. The main difference in opposite of the new architecture is low speed and circuit complexity [13]. The block diagram of one cell of this network we can see on the Fig. 6(a). Every cell contains weight register which consists from *weg_{reg}*, where is storing value of weight and *sign_{reg}*, where is information about sign of weight.

The main difference opposite to new design is, that multiplication of inputs with weights is realized sequentially. In the first step every cells inputs are multiplied with first weight and result is saved in *counter*. In the next step inputs are multiplied with second weight and result is saved in *counter*, etc... Then calculation of one's iteration of network is durable 9x15, what is 135 clock cycles.

3.2 Standard Digital CNN Network with Parallel Signed Multipliers

This network has been proposed by means of standard parallel signed multipliers. The block diagram of one cell of this network is in Fig. 6(b). Block *parallel signed multiplier* contains 9 signed multipliers, which simultaneously multiply all weights with appropriate inputs. This block contains 9 outputs corresponding with 9 results of multiplication. These results are summarized in block *adder* and followed by conversion in block *transfer function*, which realized threshold logic or hard limiter function.



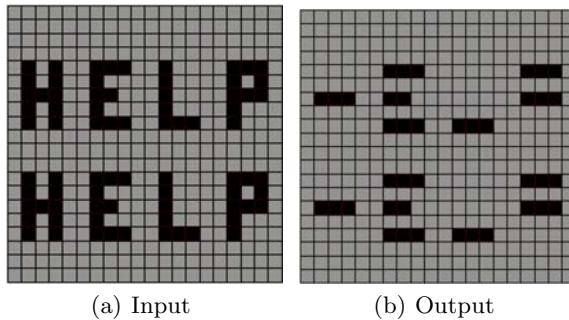
(a) The block diagram of one CNN net-work cell presented at [13]

(b) The block diagram of one cell of stan-dard CNN, which contains parallel signed multipliers.

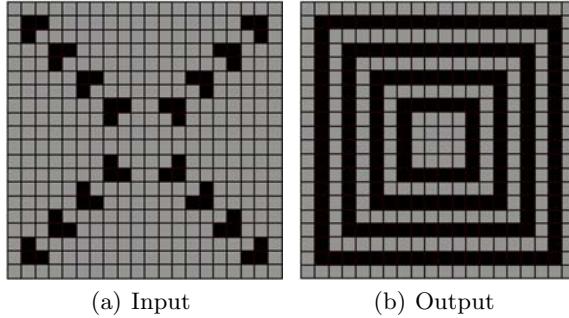
Fig. 6. The compared CNN networks

4 Results

For verification of properties and behavior for the proposed neural network we need sufficient number of cells. Proposed network is fully cascade, i.e. we are possible created network with optional number of cells. We use in our experiments applications for 400 cells. Then input of neural network is input matrix size of 20x20 and weight matrix is size of 3x3. During testing the networks we are focused mainly on network facilities as the edge detection, fill up missing edges and noise removing. During these simulations we applied as a transfer function hard-limiter. For better interpretation we displayed input to output matrix both in graphical form. Each cell is able to obtain 31 different values, which are on pictures represented by 31 shade of gray. As the first we verified network facilities by detection of horizontally edges. Input has been two words "HELP", which contained 14 horizontal edges. In Fig. 7 we can see, that required output arrive already after second iteration. Further we verify network facilities for filling up missing edges (Fig. 8). For the input to the network we are using corners. For connecting these corners the network needed 7 iterations. As last we were testing network facilities as the noise removing from picture (Fig. 9). Input has been rustle number "43" with noise cca 30%. On this figure *Output 1* we can see result



(a) Input (b) Output

Fig. 7. Detection of horizontal edges

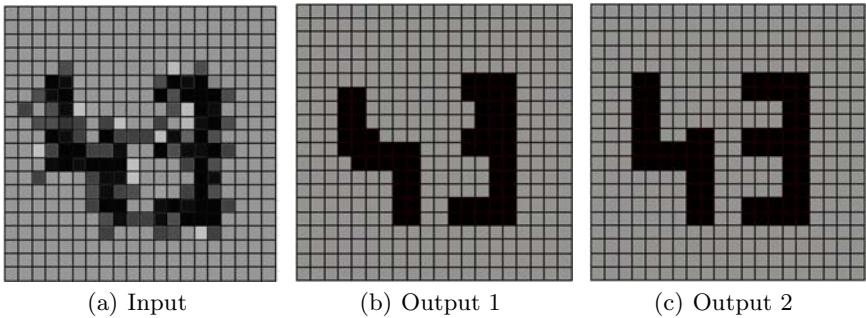
(a) Input (b) Output

Fig. 8. Completion of missing edges

of incorrect choosing weight matrix and *Output 2* result after properly choosing weight matrix. It has been 100% filtered out after first iterations.

From presented examples it is possible to conclude, that network fulfill setting requirements. Speed and correctness output is depended on properly choosing transfer function and weight matrix. Weight matrix we have defined intuitively - by the method experiment - mistake. In the future we would like to set weight matrix by learning the network.

The main aim of our work was to propose new architecture of the neural CNN network with alternative way of multiplication, that us allow to reduce chip area by implementation of this CNN network. Our main comparison parameters were the speed and size (number of gates) of network. In the Table I is comparison one's cell of the three different neural networks. Our new designed network we compared with network, which are presented in [13] and with standard network containing 5-bits parallel signed multipliers. All circuits were clocked by identical clock signal. Highest speed occurred by standard network with parallel signed multipliers, where calculation of one's iteration needs 4 clock cycles. The slowest network is our previous network, which calculation of one's iteration needs 135

**Fig. 9.** Noise removing

clock cycles. Required big number of cycles is followed by sequential multiplication inputs with weights. Our new designed network realize multiplication of all weights with corresponding inputs at once in one's time window, what allow 9 - times speed up in comparison with initially designed network. For the calculation of each iteration 15 clock cycles are needed. Speed of proposed network is then concerning standard CNN network approx. 4 - times slower. The main request of our design has been to propose CNN, which could occupy the minimum of the chip area. Parameters represented size of one's cell are in the table divided in the three columns *Bels* (Lut, mux, xor, inv,...), *Flip Flops* (flip flops) and *Total* (total number of gates). The one cell in standard network with parallel multiplier is created by 1415 number of gates. Our previous design contain 461 number of gates, what is 3 - times less than standard network. Cells of our new design network takes only 187 number of gates, which is 7.5 - times less as the standard network.

Due to parallel calculations of all cells is speed of the network with optional number of cells given as the speed of one cell calculation. The duration of one's iteration for proposed network with optional number of cells is 15 clock cycles. This speed of network is sufficient.

Design of new architecture brought saving 86.8% gates in opposite with standard CNN with parallel signed multipliers. In the Fig. 10 is dependence the consumption of gates versus number of cells.

Table 1. Comparison of parameters for CNN cell

Cell of the CNN	Parameters			
	Speed (one iteration)	Bels	Flip Flops	Total
CNN with 5-bit signed parallel multipliers	4 CLK cycles	1207	208	1415
Our previous CNN	135 CLK cycles	272	189	461
New design of the CNN	15 CLK cycles	155	32	187

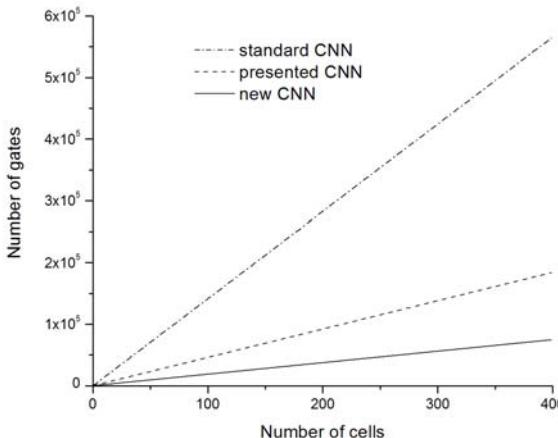


Fig. 10. Comparison of the CNN

5 Conclusion

We designed a new architecture of the digital neural network, with multiplication signals decomposed in the time by means of AND gate. Natural property of our multiplication is rounding. Values input and output from cell can be obtaining by values from -1 to 1 with step 1/15, which presented 31 shade of gray. We verified some basic properties of the proposed CNN network as are edge detection, filling of the edges and noise removing.

Our designed network is in comparison with standard CNN network with parallel signed multipliers 4 - times slower, but its relative simpler circuit which allows to spare till 86% gates and thereby allows to create network with identical parameters with fundamentally large number of cells. From the Fig. 10 we can recognize, that our new CNN needs less number of gates by the constant amount of cells, than other compared networks.

In the future we would like to create the network implementation in to FPGA chip and connected it with PC, what require create communication and user interface and define input weight matrix on base learning networks by means of incoming learning set of data.

Acknowledgments. This contribution was supported by the Ministry of Education Slovak Republic under grant VEGA No 1/0693/08 and conducted in the Centre of Excellence CENAMOST (Slovak Research and Development Agency Contract No. VVCE-0049-07).

References

1. Chua, L.O., Yang, L.: Cellular Neural Network: Theory. *IEEE Trans. Circuits Systems* 35, 1257–1272 (1988)
2. Chua, L.O., Yang, L.: Cellular Neural Network: Applications. *IEEE Trans. Circuits Systems* 35, 1273–1290 (1988)

3. Hänggi, M., Moschytz, G.S.: Cellular Neural Network: Analysis, Design and Optimization. Kluwer Academic Publisher, Boston (2000)
4. Sordo, M.: Introduction to Neural Networks in Healthcare. Open Clinical knowledge management for medical care (October 2002)
5. Larsen, J.: Introduction to Artificial Neural Network, Section for Digital Signal Processing Department of Mathematical Modeling Technical University of Denmark, 1st edn (November 1999)
6. Seung, S.: Introduction to Neural Networks: Lecture 1, The Massachusetts Institute of Technology. The Seung Lab. (September 2002)
7. Kvasnička, V., Beňušková, Ľ., Pospíchal, J., Farkaš, I., Tiňo, P., Král, A.: Úvod do teórie neurónových sietí. In: IRIS (1997) ISBN 80-88778-30-1
8. Jeon, J.: Fuzzy and Neural Network Models for Analyses of Piles, A dissertation submitted to the Graduate Faculty of North Carolina State University, Raleigh, North Carolina (August 2007)
9. Restrepo, H.F., Hoffman, R., Perez-Uribe, A., Teuscher, C., Sanchez, E.: A Networked FPGA-Based Hardware Implementation of a Neural Network Application. In: Pocek, K.L., Arnold, J.M. (eds.) Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines, FCCM 2000, Napa, California, USA, April 17-19, pp. 337–338. IEEE Computer Society, Los Alamitos (2000)
10. Muthuramalingam, A., Himavathi, S., Srinivasan, E.: Neural Network Implementation Using FPGA: Issues and Application. International Journal of Information Technology 4(2) (28-08-2008)
11. Boubaker, M., Khalifa, K.B., Girau, B., Dogui, M., Bedoui, M.H.: On-Line Arithmetic Based Reprogrammable Hardware Implementation of LVQ Neural Network for Alertness Classification. IJCSNS International Journal of Computer Science and Network Security 8(3) (March 2008)
12. Larkin, D., Kinane, A., Muresan, V., O'Connor, N.: An Efficient Hardware Architecture for a Neural Network Activation Function Generator. LNCS. Springer, Heidelberg (2006)
13. Raschman, E., Ďuračková, D.: The Novel Digital Design of a Cell for Cellular Neural Network. In: Proceedings of the Electronic Devices and Systems, Brno, September 2008, pp. 322–325 (2008)

Multifold Acceleration of Neural Network Computations Using GPU

Alexander Guzhva, Sergey Dolenko, and Igor Persiantsev

D.V. Skobeltsyn Institute of Nuclear Physics,
M.V. Lomonosov Moscow State University,
Leninskie Gory, Moscow, 119991 Russia
`nop43@rambler.ru`

Abstract. With emergence of graphics processing units (GPU) of the latest generation, it became possible to undertake neural network based computations using GPU on serially produced video display adapters. In this study, NVIDIA CUDA technology has been used to implement standard back-propagation algorithm for training multiple perceptrons simultaneously on GPU. For the problem considered, GPU-based implementation (on NVIDIA GTX 260 GPU) has lead to a 50x speed increase compared to a highly optimized CPU-based computer program, and more than 150x compared to a commercially available CPU-based software (NeuroShell 2) (AMD Athlon 64 Dual core 6000+ processor).

Keywords: GPGPU, neural networks, perceptron, NVIDIA CUDA, parallel computations.

1 Introduction

Hardware implementation of neural network algorithms is rather efficient, mostly due to highly parallel internal structure of algorithms. Usually, hardware implementation implies computations using a special neuroprocessor that supports multiple neural networks processing simultaneously and single neuron computation acceleration as well, when compared to CPU. However, using a neuroprocessor is an expensive solution; also, modern CPUs computational power obeys the well-known empirical Moore's law, according to which the productivity of CPUs doubles every year and a half. So, neuroprocessors quickly become obsolete. That is why "pure" hardware solutions for neural network computations are mostly limited with areas, where high computational speed is critical (e.g. neural network based controllers for technology, robotics and military applications).

2 GPU and GPGPU

Graphics Processing Unit (or GPU) is a kind of microprocessors used in computers for 3D graphics rendering. Initially, GPUs were used to accelerate texture based operations, such as mapping or interpolation, to implement geometric

computations in hardware. As common 3D model consists of a large number of polygons that can be processed independently in some sense, GPU architecture evolved in the direction of parallel computing.

In our days, GPUs have highly parallel architecture that allows processing numerous polygons independently. Modern GPUs belong to stream processors and imply SIMD (Single Instruction, Multiple Data) paradigm. Thus, using a modern GPU with a large number of cores can provide much greater computing speed, compared to modern CPUs.

Last generation GPUs allow running general purpose computing programs (so called GPGPU - General Purpose computation on Graphic Processing Units), not only those related to computer graphics. Developer should write program code in a special way (e.g. using compiler directives), then compile a program and load it into GPU. Usually, modern compilers use C-like language. Compilers can be easily integrated into modern program development environment, such as Microsoft Visual Studio. A number of standard libraries, such as BLAS or FFT, has been ported to be used for GPGPU-based programs. Therefore, one can say that writing a GPGPU-compliant program seems to be a “not-so-hard” task.

However, there is no unified language or technology at this moment, which could handle GPGPU-based programs regardless of GPU manufacturer (at least as far as the authors of this paper know). That’s why rewriting programs “from one GPGPU brand to another” can lead to serious difficulties. Therefore, it may happen that some complicated algorithm would have to be solely developed for a certain GPU brand.

In 2007, NVIDIA, a world-wide manufacturer of GPUs, has presented a technology named CUDA [1], which made possible creating general purpose computations oriented programs for NVIDIA GPU families.

3 Neural Networks and GPGPU

Neural networks represent a good example of how GPGPU can lead to a significant speed increase. The reason is that many of neural network architectures imply independency of neurons of each other. Thus, one can implement parallel versions of corresponding neural networks algorithms.

Historically, first MLP implementations used shader-based programs (e.g., [2][3]). Authors are sure that at present this approach is obsolete. The reason is that shaders are designed for computer graphics tasks, *not* for general purpose tasks as CUDA is. Shader-based programs had been used due to lack of available programming technologies for GPU at that time.

The authors are aware of several publications, dedicated to using neural networks with GPGPU-based programs (e.g., [4]).

In this study, we present results of program realization of multilayer perceptron (MLP) oriented CUDA-based algorithms.

How one can speed up MLP computation? The most evident way is to process a number of MLPs simultaneously. More complicated methods are strongly connected to GPU specifics, such as multiple cores or SIMD.

We shall list several general limitations we used for MLP processing, without going deeply into technological and programming details. First, we used MLPs with single hidden layer (sometimes also referred to as “three-layer MLPs”). Second, we used the so-called half-batch training mode, when several forward-propagation passes are followed by a single back-propagation pass. Third, we used simultaneous processing of multiple MLPs with the same input data, but each MLP had its own weights and its own output data.

4 Scientific Problem

CUDA-based program for processing MLPs has been designed for solving certain inverse problem related to geophysics studies. The inverse problem was to restore electrical conductivity distribution underground by the measured values of electromagnetic fields at the surface. We shall not discuss detailed physical description of the inverse problem here, as it is not significant for our technological CUDA-based solution at all. The detailed problem statement and the obtained results are discussed elsewhere [5].

The input data was represented by a matrix of floating point values (30,000 patterns by 6,552 features), and the output data was also represented by a matrix of floating point values (30,000 patterns by 336 features). This single macro inverse problem has been simplified (using several assumptions insignificant for our solution), analyzed and split into four sets each consisting of a large number of small inverse problems. We have decided to use MLPs to solve each small inverse problem.

Finally we have got 4 sets each consisting of 336 three-layer MLPs with the following properties:

- 1,648 input features
- 8 neurons in the single hidden layer with sigmoid transfer function
- 1 neuron in the output layer with sigmoid transfer function
- Learning rate has been set to 0.01 for all layers
- Momentum has been set to 0.5 for all layers
- Initial weights range has been set to ± 0.3 for all layers
- The input data for all MLPs within each set was the same
- The output data for all MLPs within each set was unique for each MLP
- Total 30,000 patterns

We have committed 5 training runs for each MLP with different initial weights randomly set each time. Thus, one can consider that we had to deal with $336 \times 5 \times 4 = 6,720$ MLPs of the specified configuration.

The input data array has been divided into training set (21,000 patterns), test set (6,000 patterns) and validation set (3,000 patterns). As usually, the training set was used for MLP training, the test set was used to prevent MLP overtraining, and the validation (out-of-sample) set was used for final MLP error estimation.

The standard back-propagation training algorithm with momentum has been implemented.

As it was mentioned, the so-called half-batch training mode was used. The MLPs were trained by groups of 10 patterns, when 10 forward-propagation passes were followed by a single back-propagation pass. To minimize the negative effect of batch training, the group was chosen randomly each time before the next batch.

The criterion of training stop was 100 epochs after minimum of error on test set.

We trained 75 MLPs simultaneously. This means that the total number of MLPs in the queue could be unlimited, but only 75 of them could be being trained at the same moment. When one of the 75 MLPs finished its training, another MLP from the queue would start its training.

We used NVIDIA CUDA technology. We will not discuss technological details, features or “special tricks”, applicable to CUDA, here (for these, please refer to CUDA documentation at NVIDIA site [1]). However, many of those tricks have been successfully used in our computer program, allowing us to gain significant speed improvements.

We should emphasize that *the achieved speed improvement benefits* are valid only for the provided MLP training general limitations and for the existing computer program, but they *do not depend on the specific values in data*. Thus, implemented program is acceptable for any case, when one should *train* multiple 3-layer MLPs with the same input data for all MLPs and unique output data for each MLP. Also, the results partly depend on the input and output data dimensions. Training parameters different from the used ones may lead to different results.

4.1 Pseudo-code for an MLP-Processing Program

```
//Procedure returns each net's weights in the
// minimal error for the test set state.

Procedure Train(N, M, DATA)
//assuming 'N' is a total number of nets to be trained
//assuming 'M' is a number of nets to be trained concurrently
//assuming 'DATA' is a container for data

Var MNow : Integer
List : Array of NN

Begin
  MNow = 0
  List is empty
  While (not all N nets trained) Do
    Begin
      While MNow < M Do
        Begin
          Add non-trained-yet net to the List
          MNow = MNow + 1
        End
    End
  End
```

```

While (not all examples shown) Do
Begin
    Select 10 random examples from DATA
    Make 10 forward propagation steps for all nets
    Make a backward propagation step for all nets
End
For Each net in the List
    If net training is complete
        Remove net from the List
        MNow = MNow - 1
    End If
    If net error for the test set is the least one
        during training process for this net
        Save net's weights as 'final' weights
    End If
Next For
End
End

```

5 Hardware for Tests

We used two different hardware configurations for testing our computer program.

System A (notebook)

- CPU: Intel Core 2 Duo T7500 2.2 GHz
- RAM: DDR2 5300 (333 MHz), 333 MHz mode
- GPU: NVIDIA GeForce 8600M GT, PCI-E x16 videocard (working in PCI-E 1.0 mode), drivers version – forceware 178.08 (*computation speed does depend on the version of driver*)
- OS: Windows Vista 32-bit

System B (desktop)

- CPU: AMD Athlon 64 x2 Dual 6000+ 3.0 GHz
- RAM: DDR2 6400 (400 MHz), 375 MHz mode
- GPU: NVIDIA GeForce GTX 260, PCI-E x16 videocard (working in PCI-E 1.0 mode), drivers version – forceware 178.08
- OS: Windows XP 32-bit

For additional information about GPU hardware, please refer to NVIDIA site [6].

6 Results

We have developed a computer program for CPU, targeted to reach maximal possible computational speed of a standard back-propagation algorithm, for comparison with GPU implementation. To test its efficiency, we have compared this CPU program with several commercially available neural network software packages.

The input data for tests consisted of 20 randomly generated features. The output data was generated as a rather simple polynomial dependence based on the values of the input variables (features). This full data set (input+output) was the same for all the programs we have tested. We have assumed that initial weights values had no significant impact on training speed performance. There were 6,000 patterns in the full data set. 60% of data were assigned to training subset; test and validation sets had their 20% and 20%.

We tried to create the same 'environment' for all the programs we have tested. So, all net training parameters, such as momentum or learning rate coefficients, were set to the same values. All data sets were also the same for all programs.

Hardware system B was used for these tests.

Results are presented in Table 1

Table 1. Comparison of several CPU-based programs

Program, hardware	Nets	Number of patterns for a batch	Full time, sec.	Time per net training epoch, sec.
NN lib	1	10	34.7	1.7
NN lib	1	1	62	3.1
Neuroshell 2	1	1	111	5.5
NeuroSolutions 5	1	1	122	6.1
MATLAB 2008a	1	1	330	16.5

Each row in Tables 1, 2 represents a separate computational experiment. The columns in Tables 1, 2 have the following sense.

"Nets" - the number of concurrently trained NNs. This value is always equal to 1 for CPU-based programs, but may vary for GPU-based programs.

"Full time, sec." - total time required for all used NNs to be trained for 20 epochs.

"Time per net training epoch, sec" - average time for 1 epoch per NN. One can compare different NN implementations time efficiency using this value.

Our program with our neural networks library (NN lib) showed very good results (compared to other programs) due to optimizations within the library development stage. Also, using half-batch mode instead of no-batch mode decreases required time approximately twice, with almost no model error increase (due to large number of patterns). Such an option does not have as large impact on CPU implementation of algorithm, as it does on GPU implementation.

Thus, one can see that GPU-based computation (Table 2, experiment 1) provided about 50 fold speed increase compared to the fastest version of CPU-based computation (experiment 5).

From practical point of view, it required about 2 months of computations using 11 CPU cores (system B, no batch training) to train 6,720 MLPs with training parameters listed above, while GPU computations on a single GTX 260 required about 17.5 hours (half-batch training). Note that this total time includes wastes

Table 2. Comparison of several GPU-based programs

Program, hardware	Nets	Number of patterns for a batch	Full time, sec.	Time per net training epoch, sec.
CUDA, B	256	10	167.5	0.033
CUDA, B	75	10	55.8	0.037
CUDA, B	64	10	51.2	0.04
CUDA, A	64	10	531.6	0.42
NN lib, B	1	10	34.7	1.73
NN lib, A	1	10	37	1.85
NN lib, B	1	1	62	3.10
NN lib, A	1	1	62.8	3.14
Neuroshell 2, B	1	1	111	5.55

due to computers shutdown for weekends, time for data loading and preparing and so on. Results seemed to be almost identical; the small difference can be explained by differences in batch training parameters.

7 Conclusion

Standard Back Propagation algorithm with momentum for training MLPs has been implemented to run on modern NVIDIA graphic processing units' family with the help of NVIDIA CUDA technology. On a serially produced video display adapter based on NVIDIA GTX 260, up to 50x speed increase compared to a highly optimized CPU program, has been obtained when training 6,720 MLPs, each having 1 hidden layer, 1,648 inputs, 8 neurons in the hidden layer and one neuron in the output layer, for a specific problem statement.

The plans of future studies include implementation of different neural architectures for GPUs, and also implementation of some specific techniques, in particular, those connected with time-series analysis.

Acknowledgments. This study has been conducted under partial financial support of the Russian Foundation for Basic Research (RFBR), grants no. 07-01-00651-a and no. 07-07-00139-a.

References

1. NVIDIA Corporation, CUDA Zone,
http://www.nvidia.com/object/cuda_home.html
2. Kyoung-Su Oh, K.S., Jung, K.: GPU implementation of neural networks. Pattern Recognition 37, 1311–1314 (2004)
3. Zhongwen, L., Hongzhi, L., Zhengping, Y., Xincai, W.: Self-Organizing Maps Computing on Graphic Process Unit. In: ESANN 2005 proceedings, European Symposium on Artificial Neural Networks Bruges, Belgium, April 27-29. d-side publi. (2005) ISBN 2-930307-05-6,
<http://www.dice.ucl.ac.be/Proceedings/esann/esannpdf/es2005-135.pdf>

4. Lahabar, S., Agrawal, P., Narayanan, P.J.: High Performance Pattern Recognition on GPU, <http://web.iiit.ac.in/~sheetal/Paper/NN.pdf>
5. Guzhva, A.G., Dolenko, S.A., Obornev, E.A., Persiantsev, I.G., Shimelevich, M.I., Shugai, J.S.: Use of Significant Feature Selection Adaptive Algorithms in Neural Networks Based Solution of the Inverse Problem of Electrical Prospecting. In: 9th International Conference Pattern Recognition and Image Analysis: New Informational Technologies (PRIA 2008), Conference Proceedings, vol. 1, pp. 215–218. Nizhni Novgorod, Russia (2008)
6. NVIDIA Corporation, <http://www.nvidia.com>

Training Recurrent Neural Network Using Multistream Extended Kalman Filter on Multicore Processor and Cuda Enabled Graphic Processor Unit

Michal Čerňanský

Faculty of Informatics and Information Technologies, STU Bratislava, Slovakia
cernansky@fiit.stuba.sk

Abstract. Recurrent neural networks are popular tools used for modeling time series. Common gradient-based algorithms are frequently used for training recurrent neural networks. On the other side approaches based on the Kalman filtration are considered to be the most appropriate general-purpose training algorithms with respect to the modeling accuracy. Their main drawbacks are high computational requirements and difficult implementation. In this work we first provide clear description of the training algorithm using simple pseudo-language. Problem with high computational requirements is addresses by performing calculation on Multicore Processor and CUDA-enabled graphic processor unit. We show that important execution time reduction can be achieved by performing computation on manycore graphic processor unit.

1 Introduction

To process data with spatio-temporal structure recurrent neural networks (RNNs) were suggested. RNNs were successfully applied in many real-life applications where processing time-dependent information was necessary. Unlike feedforward neural networks, units in RNNs are fed by activities from previous time steps through recurrent connections. In this way contextual information can be kept in units' activities, enabling RNNs to process time series. Common algorithms usually used for RNN training are based on gradient minimization of the output error. Backpropagation through time (BPTT) [1] consists of unfolding a recurrent network in time and applying the well-known backpropagation algorithm directly. Another gradient descent approach, where estimates of derivatives needed for evaluating error gradient are calculated in every time step in forward manner, is the real-time recurrent learning (RTRL) [2].

Probably the most successful training algorithms are based on the Kalman filtration (KF) [3][4][5]. The standard KF can be applied to a linear system with Gaussian noise. A nonlinear system such as RNNs with sigmoidal units can be handled by extended KF (EKF). In EKF, linearization around current working point is performed and then standard KF is applied. In case of RNNs, algorithms similar to BPTT or RTRL can be used for linearization. Methods based on the Kalman filtration often outperform common gradient-based algorithms. Multistream EKF training proved to be very successful approach for training relatively large recurrent neural networks to the complex real-life tasks from industry [6][7]. Multiple instances of the same network are trained on different data streams in the same time and coordinate weight changes are performed.

Here we address the problem of high computational requirements by using high performance of many-core processor of graphic unit. We use Compute Unified Device Architecture (CUDA) which is general purpose parallel computing architecture enabling developers to use NVIDIA's graphic processor units (GPUs) for solving complex computational problems [8]. GPUs are now highly parallel multithreaded many-core processors with high memory bandwidth capable to perform more than 10^{12} floating point operations per second. GPUs are well suited to address problems that can be expressed as data parallel computations - the same program is executed on many data elements in parallel. CUDA has been already applied in many applications such as video processing, pattern recognition or physics simulations. In [9] authors used CUDA CBLAS library for linear algebra operations of RNN EKF training.

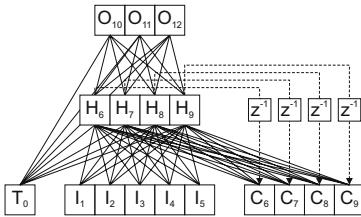
In this work we provide experiments with RNNs trained by multistream extended Kalman filter (MSEKF). This algorithm was successfully used in several real world applications and is considered to be the state-of-the-art technique for training recurrent networks. We first describe simple and elegant way of encoding recurrent neural network into data structures inspired by [10]. Then we present MSEKF in the form of simple algorithm in pseudo-language similar to Pascal. Both forward propagation of the signal and MSEKF training algorithms are given in almost copy and paste form. We discuss details of two implementations: the first using standard CPU and the second using CUDA enabled GPU. Finally we compare executions times of both implementations for different number of hidden units and different number of streams.

2 Encoding Recurrent Neural Network into Data Structures

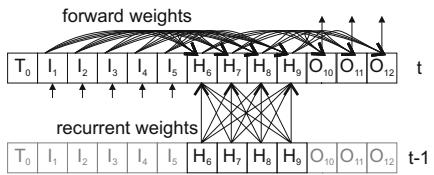
Elman's simple recurrent network (SRN) proposed in [11] is probably the most widely used RNN architecture. Context layer keeps activities of hidden (recurrent) layer from previous time step. Input layer together with context layer form extended input to the hidden layer. Elman's SRN composed of 5 input, 4 hidden a 3 output units is shown in Fig. 1a. Context units C6 to C9 hold activities of hidden units H6 to H9 from previous time step and together with input layer activities I1 to I5 they serve as extended input to the hidden units H6 to H9. All hidden and output (O10 to O12) units are connected to the special input unit T0 through threshold connections. Threshold unit T0 is set to constant value of 1.

In general, units of a neural network need not to be organized in layers. They can be randomly interconnected as soon as the directed graph representation of the network having vertices as units and edges as connections does not include any cycle. In other words, activity of a unit can be calculated knowing activity of each unit from which a weight connection exists to the given unit. Since no cyclic dependencies exist, activities of all units can be calculated. This notion also holds for RNNs, although recurrent connections form a kind of cycle in the network. But recurrent connections are sourced by activities already calculated in previous time steps and hence pose no problem in calculating actual activities. Only non-recurrent forward connections must not form a cycle in the graph representation of an RNN. Although this condition may seem to be restrictive, it is met in all commonly used feed-forward and recurrent multilayer perceptron architectures.

a) Elman's SRN - Layered Structure



b) Elman's SRN - Werbos Representation



c) Weight Connections

Weight Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	
Source Unit	wSource	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
Destination U.	wDest	6	6	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	8	8	8	8	8	8	8	8	8	8
Time Delay	wDelay	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1	1	1	1
Value	wValue																														

Weight Index	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54						
Source Unit	wSource	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
Destination U.	wDest	9	9	9	9	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	11	11	11	11	11	11	12	12	12	12	
Time Delay	wDelay	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Value	wValue																														

d) Units

Unit Index	0	1	2	3	4	5	6	7	8	9	10	11	12
First Weight	uFirstWeight												
Last Weight	uLastWeight												
Type	uType	T	I	I	I	I	I	H	H	H	O	O	O
Act. Function	uActFunc	SGM	SGM	SGM	LIN	LIN	LIN						

Fig. 1. a) Layered structure of Elman's SRN. b) Werbos representation of the Elman's SRN. c) Table gathering information on weight connections. d) Table gathering information on units.

Werbos representation of the Elman's SRN is shown in Fig. 1b. All units of the network can be indexed from 0 to $NU - 1$, where NU is the number of all units. The first unit - special input unit corresponding to the bias weight, is followed by input units, than by hidden and output units. Arranging hidden and output units is not necessary, every non-input unit can be output unit and desired activities can be specified during training phase for that unit. Unit with index i can have forward connections starting only from units with smaller indices (from index 0 to $i - 1$). Hence edges corresponding to forward weights in Fig. 1b are oriented from left to right. For every weight, indices of the source and destination unit are kept (arrays wSource and wDest), together with the time delay (wDelay) related to the weight connection. Weights can be also sorted, first by corresponding destination unit index then by connection time delay and finally by source unit index in ascending order (Fig. 1c). Forward weights have time delay of 0, what means that the actual step t activity of the source unit is fed through this connection to the destination unit. Recurrent connection have time delay t_d greater than 0 and activity from previous time step $t - t_d$ is fed through this connections. For every non-input unit the first and the last weight indices are stored (uFirstWeight, uLastWeight) together with other useful information such as unit's type (uType) and unit's activation function (uActFunc) (Fig. 1d). Usufulness of this network encoding can be seen in algorithmic description for forward propagation of the signal in custom RNN for multistream EKF (Fig. 2).

3 Multistream Extended Kalman Filter

Training of Elman's SRN, and generally any other multilayer perceptron network (recurrent or not), can be regarded as an optimal filtering [10]. The set of EKF equations for the network training can be formulated as follows:

$$\mathbf{K}(t) = \mathbf{P}(t-1)\mathbf{H}^T(t)[\mathbf{H}(t)\mathbf{P}(t-1)\mathbf{H}^T(t) + \mathbf{R}(t)]^{-1}, \quad (1)$$

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \mathbf{K}(t)\mathbf{H}(t)\mathbf{P}(t-1) + \mathbf{Q}(t), \quad (2)$$

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mathbf{K}(t)[\mathbf{D}(t) - \mathbf{O}(t)]. \quad (3)$$

Let n_w and n_o denote the number of all network weights and number of output units, respectively. \mathbf{W} is a vector of all weights of the length n_w . \mathbf{H} is the Jacobian matrix, $n_o \times n_w$, calculated in every time step and containing in rows the derivatives of corresponding output activity with respect to all weights. These derivatives can be calculated similarly to the RTRL or BPTT algorithms. \mathbf{P} is the $n_w \times n_w$ error covariance matrix, it holds error covariances corresponding to each pair of network weights. The $n_w \times n_o$ matrix \mathbf{K} called the Kalman gain is used in updating the weights \mathbf{W} according to the difference between the desired output vector \mathbf{D} and actual network output \mathbf{O} . The $n_o \times n_o$ matrix \mathbf{R} stands for the measurement noise covariance matrix and similarly to the learning rate in RTRL or BPTT can control the training speed of EKF. Note, that small process noise is still considered: the $n_w \times n_w$ matrix \mathbf{Q} stands for the process noise covariance matrix. Nonzero process noise improves convergence of the filter.

For multistream EKF training the training dataset should be split into several parts [10]. Training dataset in practical application is frequently already naturally partitioned, since it often consists of multiple subsets of different nature corresponding to different working conditions. For a chosen number of streams n_s the training sequences of the same length for each stream are selected from the different parts of the training set. For each stream the different instance of the same network is presented with the actual input-output pattern from the corresponding stream. Propagation for each stream is performed as in the case of several single networks trained on different patterns. Also derivatives of network outputs with respect to network weight are calculated in the same way as for independent networks. The procedure could follow by performing n_s independent EKF steps and by calculating overall weight changes by averaging partially calculated weight changes from each stream. But this is not the case of multistream EKF. Instead calculated derivatives of each network instance are concatenated into the single measurement update matrix $\mathbf{H}(t)$. If $\mathbf{H}_i(t)$ would be matrix of partial derivatives of output units with respect to weight for single-stream EKF training corresponding to the stream i in time step t , the matrix $\mathbf{H}(t)$ can be expressed as $\mathbf{H}(t) = (\mathbf{H}_1(t)\mathbf{H}_2(t) \dots \mathbf{H}_{n_s}(t))$. In the similar way vector of desired value $\mathbf{D}(t)$ and vector of calculated output activities $\mathbf{O}(t)$ are formed by concatenating corresponding elements, $\mathbf{D}(t) = (\mathbf{D}_1(t)^T \mathbf{D}_2(t)^T \dots \mathbf{D}_{n_s}(t)^T)^T$ and $\mathbf{O}(t) = (\mathbf{O}_1(t)^T \mathbf{O}_2(t)^T \dots \mathbf{O}_{n_s}(t)^T)^T$. Except for these changes the EKF equations remain the same (Eq. 1 to Eq. 3). Multistream EKF approach with BPTT routine for the derivative calculation is provided to make this approach clearer. First, the forward propagation of multiple streams is given in Fig. 2 and then multistream EKF training step is described in Fig. 3.

NS	- number of streams
NW	- number of weight connections
NU	- number of units (threshold unit and input units also count)
wSource[0..NW-1]	- source node indices
wDest[0..NW-1]	- destination node indices
wDelay[0..NW-1]	- connection time delays
wValue[0..NW-1]	- weight connection strengths
uFirstWeight[0..NU-1]	- indices of the first weight connection associated with the unit
uLastWeight[0..NU-1]	- indices of the last weight connection associated with the unit
uType[0..NU-1]	- unit types (THRESHOLD, INPUT, HIDDEN and OUTPUT)
uActFunc[0..NU-1]	- unit activation-function types (SGM, LIN, ...)
ACT[0..NU-1, 0..NSTEPS-1, 0..NS-1]	- all unit activities in all time steps for all streams
ACTD[0..NU-1, 0..NSTEPS-1, 0..NS-1]	- act. func. derivatives in all time steps for all streams
ActFunc(iact,actf)	- activation function calculation based on the act. func. type
ActFuncDer(iact,actf)	- derivative of the activation function
ts	- actual time step
Input(ui,ts,si)	- returns value for the input unit ui in time step ts for stream si
Output(ui,act,ts,si)	- set network output to the value act for stream si
Target(ui,ts,si)	- returns desired output unit ui activity in time step ts for stream si
1 for si=0 to NS-1 do	
2 for ui=0 to NU-1 do	
3 begin	
4 if uType[ui] = THRESHOLD then ACT[ui,ts,si] := 1.0;	
5 else if uType[ui] = INPUT then ACT[ui,ts,si] := Input(ui,ts,si);	
6 else	
7 begin	
8 iact := 0.0;	
9 for wi := uFirstWeight[ui] to uLastWeight[ui] do	
10 iact +=: wValue[wi]*ACT[wSource[wi],ts-wDelay[wi],si];	
11 ACT[ui,ts,si] := Sgm(iact);	
12 ACTD[ui,ts,si] := SgmDer(iact);	
13 end;	
14 if uType = OUTPUT then Output(ui,ACT[ui],ts,si);	
15 end;	

Fig. 2. Forward propagation for multistream EKF

Multistream forward propagation consists of NS consecutive single-stream propagations (cycle on lines 1 to 15). The first unit with index 0 correspond to the bias weight, its activity is always set to the value of 1 (line 4). Activities of input units are set to the actual input to the network (line 5). Internal activities of hidden and output units are calculated by multiplying unit's weights with source activities from corresponding time step (lines 8 to 10), operator "+:=" stand for addition of the right-hand side expression to the variable on the left-hand side. For forward weights having time delay of 0 already calculated activity from actual time step are used, for recurrent weights activities calculated in past time steps are used. Unit's activity is calculated by passing internal activity to the activation function (line 11). The derivative (line 12) calculation is only needed when gradient-based adaptation step such as backpropagation follows the forward propagation. Activities calculated for the output units are sent as the network output to the exterior (line 14).

NS	- number of streams
NO	- number of network output units
oIndex[0..NO]	- indices of output units
winSize	- unfolding window size
DO_DW[0..NO-1, 0..NW-1, 0..NS-1]	- derivatives of net. outputs with respect to all weights
DO_DNA[0..NU-1, 0..winSize-1]	- backpropagated signal
H[0..NO*NS-1, 0..NW-1]	- augmented Jacobian matrix
P[0..NW-1, 0..NW-1]	- error covariance matrix
Q[0..NW-1, 0..NW-1]	- process noise covariance matrix
R[0..NO-1, 0..NO-1]	- measurement noise covariance matrix
W[0..NW, 0..0]	- estimated vector of network weights (EKF state vector)
K[0..NW-1, 0..NO*NS-1]	- augmented Kalman gain
D[0..NO*NS-1, 0..0]	- augmented vector of desired output values
O[0..NO*NS-1, 0..0]	- augmented of calculated output values
Tr(X)	- transpose of the matrix X
Inv(X)	- inverse of the matrix X

```

1  for oui:=0 to NO-1 do for wi:=0 to NW-1 do for si:=0 to NS-1 do
2      DO_DW[oui,wi,si] := 0.0;
3
4  for si:=0 to NS-1 do
5      for oui:=0 to NO-1 do
6          begin
7              for hi:=0 to winSize-1 do for ui:=0 to NU-1 do DO_DNA[ui,hi] := 0.0;
8              DO_DNA[oIndex[oui],0] := 1.0;
9
10     for hi:=0 to winSize-1 do
11         for ui:=NU-1 downto 0 do
12             begin
13                 if uType[ui] = INPUT then break;
14                 DO_DNA[ui,hi] := DO_DNA[ui,hi]*ACTD[ui,ts-hi,si];
15                 for wi := uLastWeight[ui] downto uFirstWeight[ui] do
16                     begin
17                         if (uType[wSource[wi]] <> INPUT) AND
18                             (uType[wSource[wi]] <> THRESHOLD) AND
19                             (wDelay[wi]+hi < winSize) then
20                             DO_DNA[wSource[wi],wDelay[wi]+hi] +=:
21                                 wValue[wi]*DO_DNA[ui,hi];
22                         DO_DW[oui,wi,si] +=:
23                             DO_DNA[ui,hi]*ACT[wSource[wi],ts-hi-wDelay[wi],si];
24                     end;
25                 end;
26             end;
27
28     for wi:=0 to NW-1 do W[wi,0] := wValue[wi];
29
30     for si:=0 to NS-1 do
31         for oui:=0 to NO-1 do
32             begin
33                 D[oui+si*NS,0] := Target(ui,ts,si);
34                 O[oui+si*NS,0] := ACT[oIndex[oui],tsm,si];
35                 for wi:=0 to NW-1 do H[oui+si*NS,wi] := DO_DW[oui,wi,si];
36             end;
37
38     K := P * Tr(H) * Inv(H * P * Tr(H) + R);
39     P := P - K * H * P + Q;
40     W := W + K * (D - O);
41
42     for wi:=0 to NW-1 do wValue[wi] := W[wi,0];

```

Fig. 3. Multistream EKF with derivatives calculated by BPTT

The quantities `DO_DW` (derivatives of output activities with respect to weights) are initialized to 0 on lines 1 and 2. Then for every stream and every output unit the backpropagation through time is performed (cycle on lines 4 to 26). For one output unit the “error” signal of corresponding to this output unit is set to 1 (line 8), remaining elements of the array `DO_DNA` (derivatives of output activity with respect to the internal units’ activities) were set to 0 (line 7) and are to be calculated in the following cycle of truncated backpropagation through time (lines 10 to 25). Computation of truncated BPTT consists of backpropagating the “error” signal `winSize` time steps back (lines 10 to 25). For each unit `ui` (cycle on lines 11 to 25) and all its connections (cycle on lines 15 to 24) derivatives `DO_DNA[ui, hi]` are backpropagated through corresponding connections to the derivatives in corresponding time (lines 20 and 21). Signal is not backpropagated to the input or threshold units (condition on lines 17 and 18). The test (line 19) is also performed to ensure, that the array `DO_DNA` on lines 20 and 21 is accessed properly. Quantities `DO_DW` are built on lines 22 and 23.

Augmented arrays for desired and calculated output activities (`D` and `O`) are filled on lines 33 and 34 and the Jacobian matrix `H` on line 35. EKF update is then performed and weights are updated (line 42). In this section (lines 38 to 40) operators `*`, `+` and `-` denote matrix multiplication, addition and subtraction respectively. Since network weights are changed only by EKF part of the algorithm they do not need to be filled up in every time step and line 28 can be removed. We keep it there as a reminder than array `wValueW` and vector `W` are the same quantities.

4 Implementation Details

Time complexity of the algorithm can be easily determined from the pseudo-language description of one cycle provided in Fig. 3. The training step can be divided into two parts: first truncated BPTT is done for every stream and every output and than KF step is performed. Hence time complexity can be estimated as:

$$T_{\text{MSEKF}} = T_{\text{BPTT}} + T_{\text{KF}} = O(n_s \times n_o \times n_w \times h) + O(n_o \times n_w^2). \quad (4)$$

Since for huge networks number of streams n_s , number of output units n_o and the truncated BPTT window size h are smaller than the number of weights n_w , the time complexity of MSEKF can be expressed as:

$$O_{\text{MSEKF}} \approx O(n_w^2). \quad (5)$$

Although in practice multiplicative constants (not revealed in big-O notation) do matter we can already see that the computation is dominated by the time complexity of the KF part.

Standard way how implement operations with matrices is the usage of some linear algebra package. All major processor vendors offer high-speed parallel multithreaded versions of BLAS (Basic Linear Algebra Subprograms). Intels solution is MKL (Math Kernel Library) offering BLAS and LAPACK (Linear Algebra PACKAGE) interfaces. Other choices are ACML (AMD Core Math Library) tailored for AMD processors and GotoBLAS library which is probably the fastest BLAS library available. Since the target

platform is a computer equipped with Intel Core i7 quad core processor and GotoBLAS has not yet been optimized for Core i7 architecture we have chosen MKL as a linear algebra package for CPU implementation of MSEKF.

CUDA platform contains CUBLAS (CUDA BLAS) library benefiting from parallelism of many-core GPU architecture but providing standard BLAS application interface. Whole KF part was implemented to be performed on the GPU including implementation of cholesky solver used for finding Kalman gain K . Only vector $\mathbf{E} = \mathbf{D} - \mathbf{O}$ and matrix \mathbf{H} are transferred into the GPU device and weight vector \mathbf{W} is transferred back from the CUDA device.

Other source of parallelism is the truncated BPTT step since iterations are independent. $n_o \times n_s$ iterations are performed in the truncated BPTT part of the MSEKF step. The only modification is that NO_DNA array cannot be shared between iterations executed in parallel. There are several programming models that can be used to improve performance of this part of the algorithm by splitting computation into multiple threads. We have chosen Intel's Threading Building Blocks (TBBs) and "parallel.for" construct was used in straightforward way. Implementation of BPTT step using CUDA is the task for near future.

5 Results

Tests for both CPU and GPU implementations of MSEKF were conducted on the same machine running Microsoft Windows XP SP 3 equipped with Intel Core i7 Nehalem processor operating at 2.67GHz, 3GB of RAM and Nvidia GeForce N280 graphic card. Intels hyperthreading technology was turned off in BIOS as recommended in MKL manual.

We performed tests with Elman's simple recurrent network trained for the next value prediction of Santa Fe laser sequence. Since RNNs are also frequently used by cognitive science community for modeling symbolic sequences the second dataset was generated by Elman's grammar [12]. In this case the networks were trained on the next symbol prediction task. The length of both sequences was 10000 values, while training on Laser sequence RNNs has single input and output unit, for Elman datasets RNN's input and output layer consist of 24 units.

We present simulation results for various number of streams and number of hidden units. MSEKF training run times in seconds are summarized in the Tab. 1 and correspond to one epoch - one presentation of the training sequence. For a given number of streams n_s the training sequence is divided into n_s parts of the same length and the network performs $10000/n_s$ MSEKF training cycles per epoch. Hence higher number of streams does not mean that more operations were performed during training.

We provide results for computation performed in single precision only for two reasons. First using double precision did not bring any difference considering resulting performance (similarly to [9]). We also encountered no problems with numerical stability. The second reason is that GPUs performance in double precision is lower since graphic hardware is optimized for single precision computation. For applications where numerical precision is crucial performing computation on standard multicore processor may be a better choice.

Table 1. Simulations times in seconds for BPTT and EKF parts of CPU and CUDA implementations of MSEKF

LASER	BPTT - no TBB	BPTT - TBB	EKF - CPU	EKF - CUDA
MSEKF-16HU-1S	0.001	0.001	1.01	5.02
MSEKF-16HU-4S	0.25	0.004	0.51	1.51
MSEKF-16HU-16S	0.18	0.065	0.52	0.76
MSEKF-16HU-64S	0.17	0.045	0.89	0.76
MSEKF-64HU-1S	2.00	2.02	210.51	124.87
MSEKF-64HU-4S	2.25	0.51	60.16	33.63
MSEKF-64HU-16S	2.18	0.56	20.67	11.29
MSEKF-64HU-64S	2.25	0.55	21.00	8.84
ELMAN	BPTT - no TBB	BPTT - TBB	EKF - CPU	EKF - CUDA
MSEKF-16HU-1S	11.00	3.01	46.51	35.59
MSEKF-16HU-2S	10.99	3.06	67.18	37.63
MSEKF-16HU-4S	11.49	3.02	111.74	69.21
MSEKF-16HU-8S	11.72	2.92	205.48	120.62
MSEKF-32HU-1S	27.99	7.15	184.08	95.12
MSEKF-32HU-2S	27.99	7.67	217.80	104.51
MSEKF-32HU-4S	29.00	7.42	322.48	177.40
MSEKF-32HU-8S	29.32	7.32	548.07	319.98

We provide results for both unparallelized (no TBB) and parallelized (TBB) BPTT part. Please note that the time requirements of BPTT part are the same for fixed number of streams, since the same number of backward propagations through time is performed for one epoch. Unsurprisingly parallelized version of the BPTT takes much less time than its unparallelized counterpart. Almost 4 times better performance was achieved, hence the performance of this part scales well with the number of cores.

As can be seen from Tab. II significant run time reduction for larger networks can be obtained by performing MSEKF training on CUDA-enabled GPU. On the other side standard CPU performs well for smaller networks since massive parallelism of many-core GPUs is not used. In general smaller networks benefit neither from multi-core nor many core processors because of small level of parallelism when doing calculations with small matrices.

6 Conclusion

Multistream extended Kalman filter is probably the most successful algorithm for training recurrent neural networks. The main drawback of MSEKF is sever computational requirements in comparing with common approaches such as BPTT or RTRL. This usually prevents thorough search for better parameters or other experimentation when performing model selection. In this paper we first provide detailed description of the algorithm using simple pseudo-language. Algorithm is almost in a copy and paste form. Then we provide results of implementations targeting CPU and CUDA-enabled GPU platforms. We show that significant reduction of execution time can be achieved by performing calculations on graphical processing units when training large networks.

Acknowledgments. This work was supported by the grants Vega 1/0848/08, Vega 1/0822/08 and APVV-20-030204.

References

1. Werbos, P.: Backpropagation through time; what it does and how to do it. Proceedings of the IEEE 78, 1550–1560 (1990)
2. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation 1, 270–280 (1989)
3. Williams, R.J.: Some observations on the use of the extended Kalman filter as a recurrent network learning algorithm. Technical Report NU-CCS-92-1, Northeastern University, College of Computer Science, Boston, MA (1992)
4. Čerňanský, M., Beňušková, Ľ.: Simple recurrent network trained by RTRL and extended Kalman filter algorithms. Neural Network World 13(3), 223–234 (2003)
5. Trebatický, P.: Recurrent neural network training with the kalman filter-based techniques. Neural network world 15(5), 471–488 (2005)
6. Feldkamp, L., Prokhorov, D., Eagen, C., Yuan, F.: Enhanced multi-stream Kalman filter training for recurrent networks. In: Suykens, J., Vandewalle, J. (eds.) Nonlinear Modeling: Advanced Black-Box Techniques, pp. 29–53. Kluwer Academic Publishers, Dordrecht (1998)
7. Prokhorov, D.V.: Toyota prius hev neurocontrol and diagnostics. Neural Networks 21, 458–465 (2008)
8. NVIDIA: NVIDIA CUDA programming guide. Technical report (2008)
9. Trebatický, P.: Neural network training with extended kalman filter using graphics processing unit. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 198–207. Springer, Heidelberg (2008)
10. Prokhorov, D.V.: Kalman filter training of neural networks: Methodology and applications. In: Tutorial on IJCNN 2004, Budapest, Hungary (2004)
11. Elman, J.L.: Finding structure in time. Cognitive Science 14, 179–211 (1990)
12. Elman, J.: Distributed representations, simple recurrent networks, and grammatical structure. Machine Learning 7, 195–225 (1991)

A Non-subtraction Configuration of Self-similitude Architecture for Multiple-Resolution Edge-Filtering CMOS Image Sensor

Norihiro Takahashi and Tadashi Shibata

Department of (Frontier Informatics;
Electrical Engineering and Information Systems),
The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
ntaka@else.k.u-tokyo.ac.jp, shibata@ee.t.u-tokyo.ac.jp
<http://www.if.t.u-tokyo.ac.jp/>

Abstract. The *self-similitude* architecture developed in our previous work for multiple-resolution image perception [1] has been transformed into a non-subtraction configuration. In contrast to the previous work, the subtraction operations are entirely eliminated from the computation repertory of processing elements. As a result, the hardware organization of multiple-resolution edge-filtering image sensor has been greatly simplified. In addition, a fully pixel-parallel *self-similitude* processing has been established without any complexity in the interconnects. A proof-of-concept chip capable of performing four directional edge filtering at full, half and quarter resolutions was designed in a $0.18\mu\text{m}$ 5-metal CMOS technology and was sent to fabrication. The performance was verified by circuit simulation (Synopsys NanoSim), showing that the four directional edge filtering at multiple resolutions is carried out at more than 1000 frames/sec. with a clock rate of 500kHz.

Keywords: Edge Detection, Hardware Implementation, Image Perception, Multiple-Resolution, Self-Similitude.

1 Introduction

Edge detection is one of the most fundamental image pre-processing for human-like visual perception systems. It is grounded on the fact that human image recognition relies on edge information in various orientations extracted from input images [2]. In fact, many algorithms have been established using directional edge detection in the fields of feature extraction from images and image feature vector representation [3]-[4].

The multiple-resolution image processing scheme is exploited to handle various sizes of images. When the edge-based image representation [4] is combined with multiple-resolution processing, image recognition robust against scaling has

been demonstrated [5]. However, because directional edge filtering in multiple-resolutions is computationally very expensive, hardware implementation is necessary for real-time recognition system. In this regard, the *self-similitude* architecture was proposed for the first time and implemented in an analog CMOS image sensor in our previous work [1].

The concept of the multiple-resolution image perception is illustrated in Fig. 1 [5], taking face detection as an example. There are three different-size facial images in the full size picture, in which only the smallest size face is enclosed in the detection window. However, by scaling the picture with $1/2$, $1/4$... resolutions, different size faces are all detected. It was shown in [5], where any size facial images can be correctly recognized by the combination of $(1/2)^n$ -scaling of input images and the use of three different sizes for template images, viz. 100%, 80%, and 60%.

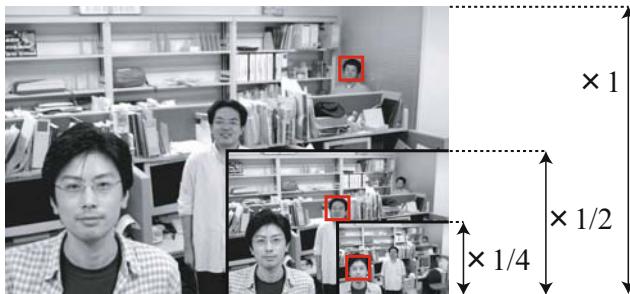


Fig. 1. Concept of multi-resolution image perception. Any size facial images can be detected by combination of $(1/2)^n$ -scaling of input images and use of three template image sizes of 100%, 80%, and 60%.

Multiple-resolution image processing functions were utilized in various objectives and already implemented in several chips. A multiresolution image sensor [6] has the simple function of multiple-resolution image readout. There were also image sensors for programmable kernel filtering, where the kernel size is variable and any sort of convolution is practicable [7-9]. An edge detection image sensor was reported employing the multi-scale veto algorithm [10]. In a spatial-temporal multiresolution image sensor [11], the resolution is intentionally changed to realize low-power object tracking. Such a spatial-temporal multiple-resolution function was first implemented in [12].

In our previous work [1], a multiple-resolution edge-filtering CMOS image sensor was developed based on the *self-similitude* architecture. The proof-of-concept chip was designed using the voltage mode analog circuitry and fabricated to verify the concept. However, the subtraction operation included in the basic repertory of processing elements (PEs) made the PE control very complicated and reduced the fill factor of each pixel. In order to solve the problem, one row (and one column) of subtraction circuitries are placed at the peripheries of

the photodiode and PE array, and the subtraction operation was removed from the PE array. However, this has reduced the advantage of the *self-similitude* architecture which offers a fully pixel-parallel processing capability of a CMOS image sensor for multiple-resolution filtering.

In this work, a non-subtraction configuration of the *self-similitude* architecture was proposed in contrast to the subtraction-separated configuration in our previous work [1]. By assigning a plus or minus sign to the pixel value readout from each pixel, all computations have been achieved using only additions. As a result, the hardware organization and the PE control have been simplified in a great deal and the merit of the fully pixel-parallel computation has been enabled in this configuration. This allows us to develop the application of *self-similitude* architecture not only to edge-filtering, but also to different types of multiple-resolution image processing on CMOS image sensors.

2 Non-subtraction Configuration of Self-similitude Algorithm

The basic idea of the *self-similitude* algorithm for multiple-resolution edge filtering was already presented in our previous paper [1]. However, because the algorithm has been modified to remove subtraction operations, a brief explanation is given in the following. Fig. 2(a) shows the MIMD (multiple instruction multiple data) organization of the multiple-resolution edge filter. Photodiodes (PDs) are located at four corners of each PE, allowing each PE to receive four pixel luminance data using only intra-PE interconnects [13]. Fig. 2(b) shows the output signal from a PD cell which bears a plus or minus sign with respect to the bias value.

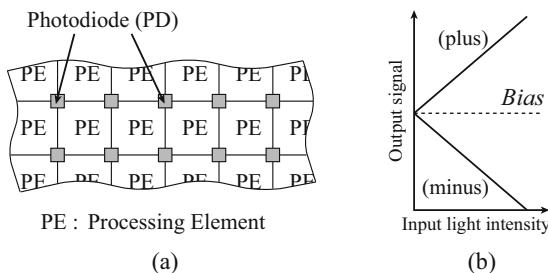


Fig. 2. MIMD organization of multi-resolution edge-filtering image sensor (a) and output signal from a PD cell (b)

Fig. 3 compares the fundamental operation repertory in the modified algorithm with that of the previous one. In the modified algorithm, only two types of operations, addition of the inputs from PDs at four corners and addition of

the inputs from four neighboring PEs, are defined. Because the number of operations is reduced from ten to two, the control signal management of PEs has been greatly simplified. Furthermore, because the operation necessary for edge filtering is only addition, the PEs for subtraction which were placed at the periphery of the PD and PE array in the previous work have been entirely eliminated. The sign patterns of convolution kernels are generated by assigning a proper plus/minus sign to each PD readout.

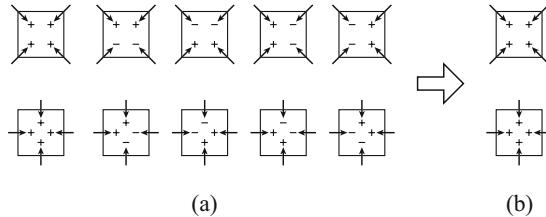


Fig. 3. Fundamental operation repertory in previous work (a) and in this modified algorithm (b)

The procedure of multiple-resolution directional edge filtering is explained in Fig. 4 using two typical examples. Full-resolution horizontal edge filtering proceeds in two steps. Before the processing, plus signs are assigned to upper PD cells, while minus signs are assigned to lower PD cells as shown in the left of Fig. 4 (a). In the first step, four PEs carry out the diagonal-input addition. In the second step, the center PE carries out orthogonal-input addition using the first step results (R). This results in a horizontal edge filtering of the 4x4 kernel. During the processing, the same operations are being carried out in other equivalent locations simultaneously. Then the roles (first step operations and second step operations) of respective PEs are interchanged and the same procedure is repeated to accomplish the filtering for the entire image.

The half-resolution +45-degree edge filtering requires three steps as illustrated in Fig. 4 (b). Before the processing, PD cells in the upper left triangle region are assigned with plus signs, PD cells in the lower right triangle region with minus signs, and PD cells on the central diagonal line with zero values. 16 PEs (+ marked) carry out the diagonal addition in the first step and four PEs (+ marked) carry out the orthogonal addition in the second step using the first-step operation results (R). In the third step, the center PE (also + marked) conducts the addition using the second-step operation results (R'). The resultant kernel is for an 8x8, +45-degree edge filtering. In order to expand the kernel region to one step a larger area, the orthogonal addition need be repeated one more time.

In the previous work, a filtering kernel pattern having plus and minus coefficients was produced by addition and subtraction operations of PEs. This increased the number of operations of a PE as shown in Fig. 3 (a). In the present work, however, plus and minus signs are assigned to output signals from PDs to

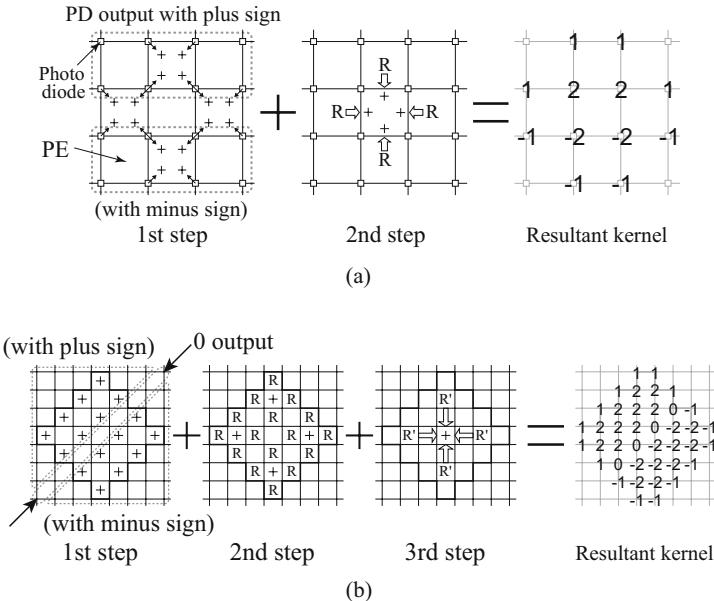


Fig. 4. Full-resolution horizontal (a) and half-resolution +45-degree (b) edge filtering operations

produce kernel patterns. Then, only addition operations are required for PEs. This has resulted in a great simplification of the hardware organization and a fully pixel-parallel implementation of the *self-similitude* architecture in the CMOS image sensor has been enabled for various kinds of filtering processing in the multiple-resolution regime.

3 Hardware Organization of Edge Filtering Image Sensor

The hardware organization in this work is shown in Fig. 5 in comparison with that of the previous work. The hardware consists of a 56x56-PD array and a 55x55-PE array. Interconnects are provided among all nearest neighbor PEs for full resolution processing, among every two and four rows/columns for half and quarter resolution processing, respectively. As compared to the previous work, the PEs for subtraction are removed in this work, which has greatly simplified the control signal management for PEs and also has reduced the chip area.

Each PD cell has the function of linearly transforming the light intensity into the analog electrical signal. The output of each PD is controlled by the external digital signals. That is, the output value (*Out*) is plus/minus signed value of input signal (*In*) with bias value (*Bias*), or just bias value itself.

$$Out = \pm In + Bias \text{ (Plus/Minus), Bias (Zero).} \quad (1)$$

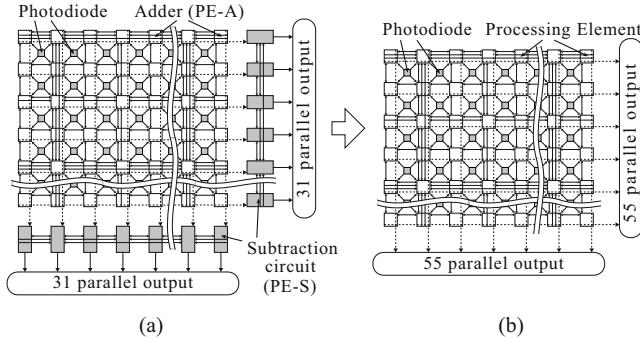


Fig. 5. Hardware organization of multi-resolution directional edge filtering image sensor in previous work (a) and in present work (b)

Fig. 6 represents the block diagram of PE cell. One of the four input data (from PD, nearest/second/fourth neighbor PEs) is selected by the selector at each side of the PE. Then, the addition of these selected inputs is carried out and the result is sent to neighbor PEs in all directions.

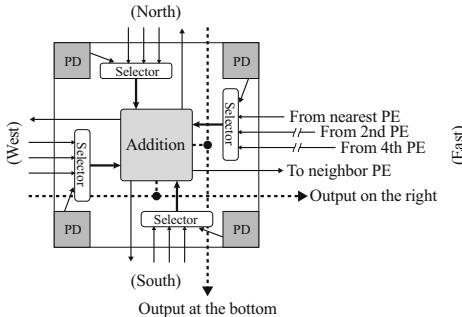


Fig. 6. Block diagram of PE cell

Examples of single row/column parallel processing are illustrated in Fig. 7. An example of full-resolution horizontal edge filtering is shown in Fig. 7(a), where the second step of the processing is carried out in every two PEs (depicted in gray) in the same row. An example of full-resolution +45-degree edge filtering is represented in Fig. 7(b), and the second step is operated in the PEs on the diagonal line. Fig. 8 shows the pixel-parallel processing using the same examples of the processing in Fig. 7. Control signals for assigning plus and minus signs are inputted from the outside of the PD array. If the control signals are made to propagate in four directions in the PD array, simple wiring scheme would be achieved even for pixel-parallel processing.

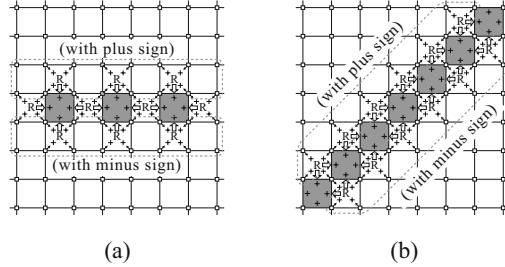


Fig. 7. Examples of row/column parallel processing: full-resolution horizontal edge filtering (a) and $+45$ -degree edge filtering (b)

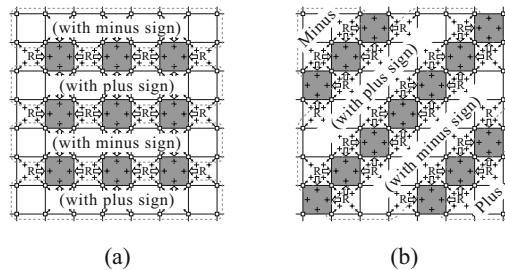
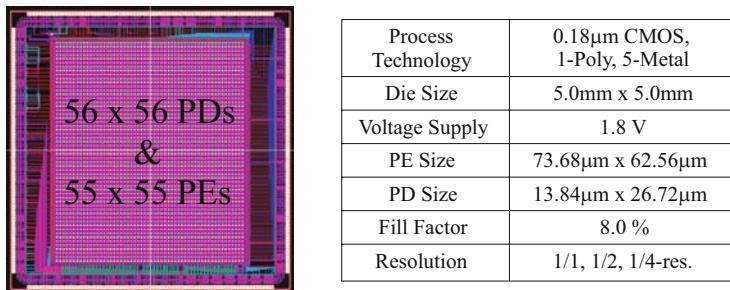
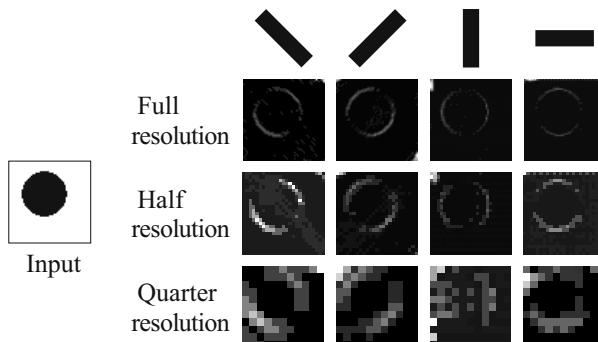
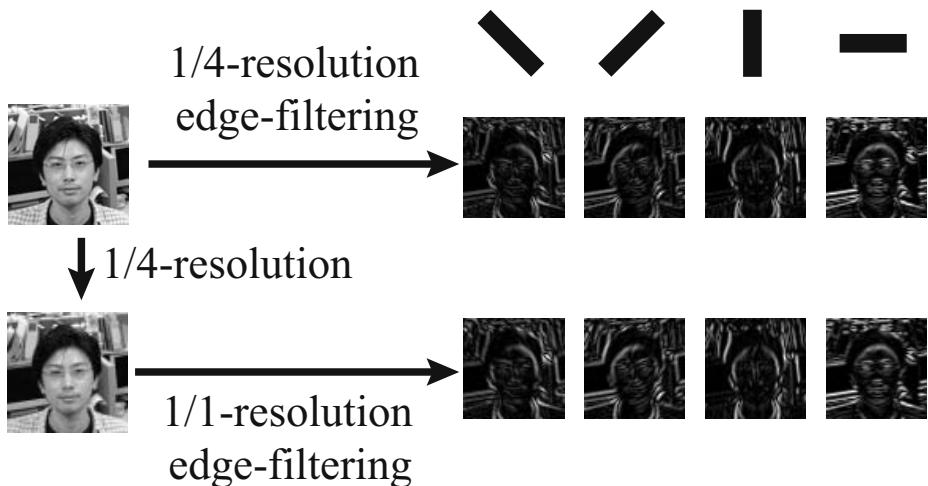


Fig. 8. Examples of pixel-parallel processing: full-resolution horizontal edge filtering (a) and $+45$ -degree edge filtering (b)

4 Hardware Implementation and Discussions

The concept of the non-subtraction configuration of the *self-similitude* architecture was implemented in the hardware using current-mode analog computation. The layout and the specification of the proof-of-concept chip are shown in Fig. 9. In this chip, only the line parallel processing as illustrated in Fig. 7 was implemented for the purpose of verifying the concept. However, extension to the pixel-parallel processing shown in Fig. 8 can be easily accomplished by just changing the control circuitries outside the PD array so that they produce the signals necessary for pixel-parallel processing.

Simulation results are demonstrated in Fig. 10. The illumination data for the PDs were virtually given as external analog signals. A circular picture was used as an input. Circular silhouettes corresponding to respective edge directions are seen in the edge filtering results. In this simulation, the execution time for each row/column edge filtering was within $2\mu s$, which implies every directional edge filtering at multiple-resolution for entire image is accomplished in less than $220\mu s$. This result means the presented multiple-resolution edge filtering processor has the capability of performing more than 1000 frames/sec (limited by the PD integration and readout time). Since the power consumption is easily reduced by introducing the enable function, the chip is also embeddable in low power systems.

**Fig. 9.** Layout and specification of proof-of-concept chip**Fig. 10.** Simulation results of multi-resolution directional edge filtering**Fig. 11.** Verification of *self-similitude* computation algorithm

Because the edge filtering kernels of lower resolutions are not the exact expansions of the full-resolution kernels, the *self-similitude* algorithm is verified by software simulation. Fig. II shows the simulation results of quarter-resolution directional edge filtering using *self-similitude* algorithm and full-resolution directional edge filtering after lowering the resolution from full to quarter. The simulation results indicate that these two different procedures of quarter-resolution edge filtering generate almost the same images. That is, the tiny kernel differences between different resolutions have no great influence upon the final edge pictures.

5 Conclusion

A multiple-resolution directional edge filtering processor has been developed based on the *self-similitude* architecture without using subtraction function. As a result, the hardware structure and the controlling method is greatly simplified. The proof-of-concept chip was designed and the concept has been verified by simulations.

Acknowledgments

The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), The University of Tokyo in collaboration with Rohm Corporation and Toppan Printing Corporation. The work is partially supported by the Ministry of Education, Science, Sports, and Culture under Grant-in-Aid for Scientific Research (No. 17206030).

References

1. Takahashi, N., Fujita, K., Shibata, T.: An Analog Self-Similitude Edge-Filtering Processor for Multiple-Resolution Image Perception. In: 2008 IEEE International Symposium on Circuits and Systems, pp. 1640–1643 (2008)
2. Hubel, D.H., Wiesel, T.N.: Receptive fields of single neurons in the cat's striate cortex. *J. Physiology* 148, 574–591 (1959)
3. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. *J. Computer Vision* 60(2), 91–110 (2004)
4. Yagi, M., Shibata, T.: An Image Representation Algorithm Compatible with Neural-Associative-Processor-Based Hardware Recognition Systems. *Trans. Neural Networks* 14(5), 1144–1161 (2003)
5. Suzuki, Y., Shibata, T.: An Edge-Based Face Detection Algorithm Robust Against Illumination, Focus, and Scale Variations. In: 13th European Signal Processing Conference, pp. 2279–2282 (2004)
6. Kemeny, S.E., Panicacci, R., Pain, B., Matthies, L., Fossum, E.R.: Multiresolution image sensor. *Trans. Circuits and Systems for Video Technology* 7(4) (1997)
7. Cembrano, G.L., et al.: A 1000 FPS at 128 × 128 vision processor with 8-bit digitized I/O. *J. Solid-State Circuits* 39(7) (2004)

8. Dudek, P., Hicks, P.J.: A general-purpose processor-per-pixel analog SIMD vision chip. *Trans. Circuits and Systems* 52(1) (2005)
9. Olyaei, A., Genov, R.: Focal-Plane Spatially Oversampling CMOS Image Compression Sensor. *Trans. Circuits and Systems I* 54(1), 26–34 (2007)
10. McIlrath, L.D.: A CCD/CMOS focal-plane array edge detection processor implementing the multiscale veto algorithm. *J. Solid-State Circuits* 31(9), 1239–1247 (1996)
11. Choi, J., Han, S.W., Kim, S.J., Chang, S.I., Yoon, E.: A Spatial-Temporal Multiresolution CMOS Image Sensor With Adaptive Frame Rates for Tracking the Moving Objects in Region-of-Interest and Suppressing Motion Blur. *J. Solid-State Circuits* 42(12), 2978–2989 (2007)
12. Gruev, V., Etienne-Cummings, R.: Implementation of Steerable Spatiotemporal Image Filters on the Focal Plane. *Trans. Circuits and Systems II* 49(4), 233–244 (2002)
13. Nakashita, Y., Mita, Y., Shibata, T.: An analog visual pre-processing processor employing cyclic line access in only-nearest-neighbor-interconnects architecture. In: 2005 NIPS Conference (2005)

Current-Mode Computation with Noise in a Scalable and Programmable Probabilistic Neural VLSI System

Chih-Cheng Lu and H. Chen

The Dept. of Electrical Engineering,
The National Tsing-Hua University, Hsin-Chu, Taiwan 30013
hchen@ee.nthu.edu.tw

Abstract. This paper presents the VLSI implementation of a scalable and programmable Continuous Restricted Boltzmann Machine (CRBM), a probabilistic model proved useful for recognising biomedical data. Each single-chip system contains 10 stochastic neurons and 25 adaptable connections. The scalability allows the network size to be expanded by interconnecting multiple chips, and the programmability allows all parameters to be set and refreshed to optimum values. In addition, current-mode computation is employed to increase dynamic ranges of signals, and a noise generator is included to induce continuous-valued stochasticity on chip. The circuit design and corresponding measurement results are described and discussed.

Keywords: Probabilistic VLSI, noise, scalable and programmable systems.

1 Introduction

Probabilistic models use stochasticity to generalise the natural variability of data, and have been shown promising for reasoning biomedical data or for solving weakly-constrained problems such as pattern recognition. Realising probabilistic models in the Very-Large-Scale-Integration (VLSI) is thus attractive for the application like intelligent sensor fusion in implantable devices [1] [2]. However, only a few probabilistic models are amenable to VLSI implementation [3] [4], and most of which relies greatly on precise computation of Bayesian rules or vector products, which becomes infeasible as transistor noise and hardware non-ideality grow.

The CRBM is a probabilistic model which has been shown capable of classifying biomedical data reliably [5] and has been realised as a probabilistic VLSI system [6], potential for being an intelligent embedded system in implantable devices. With a fixed number (six) of neurons, however, the prototype system is limited to model two-dimensional data, while biomedical signals in real-world applications are normally high-dimensional and complex. Therefore, modular design is employed in the VLSI implementation presented here, allowing the network size to be expanded by connecting multiple chips. All parameters of the system are stored in dynamic analogue memory which can be not only refreshed at optimum values reliably but also trained by chip-in-a-loop configuration. The full system has been designed and fabricated with the TSMC

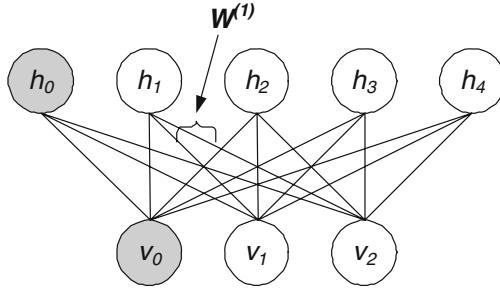


Fig. 1. The architecture of a CRBM model with two visible and four hidden neurons. v_0 and h_0 represent biasing units with invariant outputs $v_0 = h_0 = 1$.

0.35 μ m CMOS technology. Following a brief introduction of the CRBM model, the architecture of the VLSI system, the circuit design, and the measurement results will be presented.

2 The CRBM Model

The CRBM consists of one visible and one hidden layers of stochastic neurons with inter-layer connections only, as shown in Fig. 1. The number of visible neurons corresponds to the dimension of data, while that of hidden neurons is chosen according to data complexity [5]. Let w_{ij} represents the bi-directional connection between v_i and h_j . The stochastic behaviour of a neuron s_i is described by [5]

$$s_i = \varphi_i(a_i \cdot (\sum_j w_{ij} \cdot s_j + N_i(0, \sigma))) \quad (1)$$

where $N_i(0, \sigma)$ represents a zero-mean Gaussian noise with variance σ^2 , and $\varphi_i(\cdot)$ a sigmoid function with asymptotes at ± 1 (e.g. $\tanh(\cdot)$). Parameter a_i controls the slope of the sigmoid function and thus the variance of s_i .

As a generative model, the CRBM learns to "regenerate" the probabilistic distribution of training data at its visible neurons. Testing data can be subsequently classified according to the responses of hidden neurons [5]. Both $\{a_i\}$ and $\{w_{ij}\}$ can be trained by the simplified minimising-contrastive-divergence (MCD) algorithm, requiring only addition and multiplication of neurons' states to determine updating direction [7]. The simplicity and the locality make the training algorithm hardware-friendly.

3 System Architecture

Fig. 2 shows the architecture of the scalable and programmable CRBM system [8], containing neuron modules (v_i and h_j), synapse modules (w_{ij}), a noise generator, and digital control circuits. The refreshing unit is designed to be realised by a microcontroller off-chip. Each synapse module contains two multipliers to calculate $w_{ij}h_j$ and $w_{ij}v_i$ as current inputs for neurons v_i and h_j , respectively. Each neuron $v_i(h_j)$ then sums up the

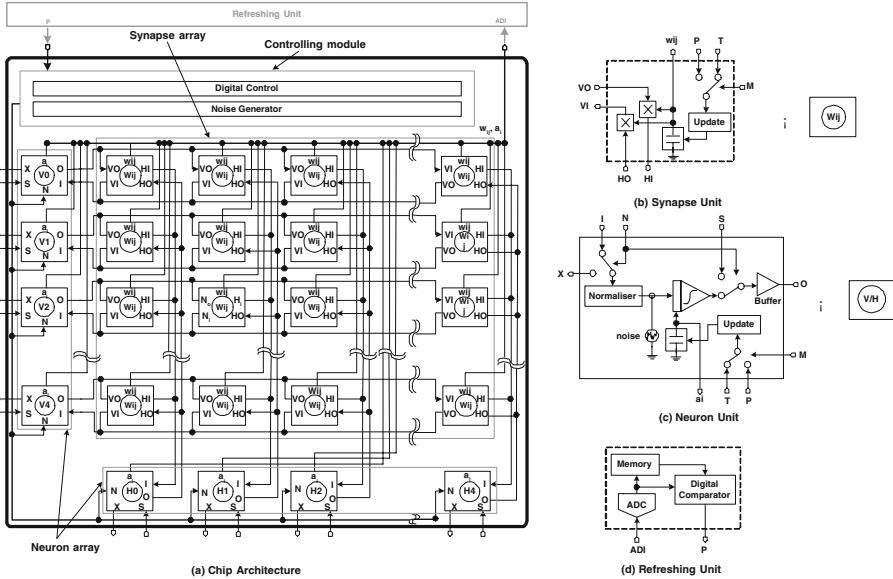


Fig. 2. The architecture of a scalable and programmable CRBM system and its functional units

currents on the same row (column) at the terminal I (Fig. 2(c)), and passes the total current through sigmoid function to generate an output voltage at terminal O. In addition, each neuron includes a noise input to makes its output probabilistic.

The modular design enables the CRBM system to expand its network size easily by interconnecting multiple chips. For example, an $M \times N$ chip array forms a CRBM system with $5M$ visible and $5N$ hidden neurons. Synapse modules in the same row (column) transmit output currents to the left- (bottom-) most neurons in the row (column). Each neuron module $v_i(h_j)$ then transmits voltage output back to synapse modules in the same row (column). The control signal N in each neuron (Fig. 2(c)) determines whether the neuron is enabled. When $N=1$, current inputs at terminal I are passed through sigmoid circuit to generate the neuron's output at terminal O. When $N=0$, the current inputs at terminal I are simply directed to terminal X, and the neuron output is buffered from terminal S into terminal O. A current normaliser is included to avoid the saturation of sigmoid circuit.

The parameters $\{w_{ij}\}$ and $\{a_i\}$ are stored locally as voltages across capacitors in the synapse and neuron module, respectively. The updating circuit employed in Fig. 2(b) is used to tune the capacitor voltages with infinitely small steps according to the digital input P or T. In training mode ($M=1$), the digital signal T is selected and calculated according to the simplified MCD algorithm. As soon as optimum levels are obtained, the analogue-to-digital converter (ADC) in the refreshing unit stores w_{ij} and a_i into digital memory (Fig. 2(d)). Note that one ADC can be shared by all parameters. In refreshing mode ($M=0$), parameter values on capacitors are sampled periodically by the ADC, compared with optimum levels stored in the memory, and updated according to the output P of digital comparator.

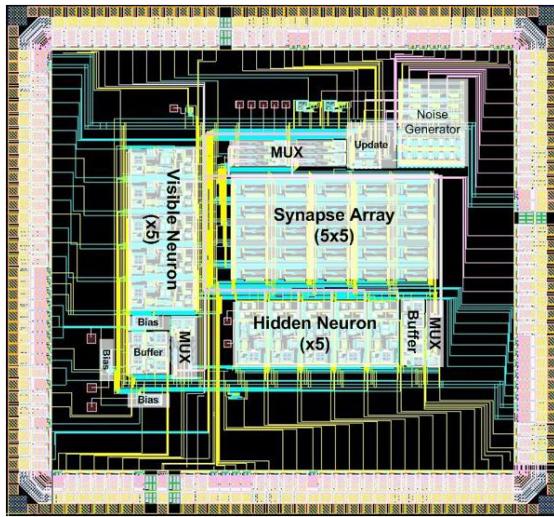


Fig. 3. The layout of the scalable and programmable CRBM system in VLSI

Table 1. The mapping of parameter values between software simulation and hardware implementation

	Matlab	VLSI(V)
s_i	$[-1.0, 1.0]$	$[1.0, 2.0]$
w_{ij}	$[-3.0, 3.0]$	$[0.0, 3.0]$
a_i	$[0.5, 5.0]$	$[1.0, 2.5]$

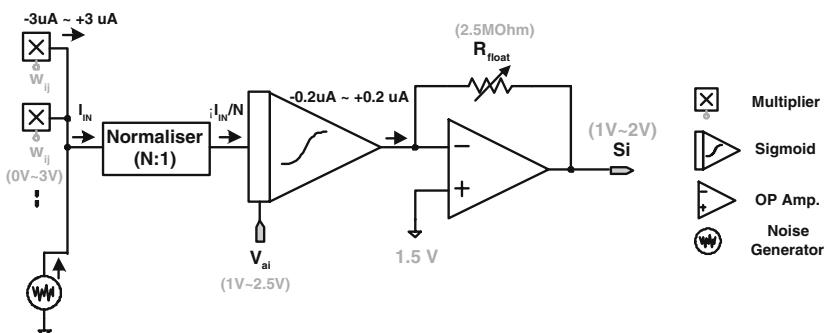


Fig. 4. The block diagram of the neuron with corresponding signal flows and the dynamic ranges of parameters in each stage

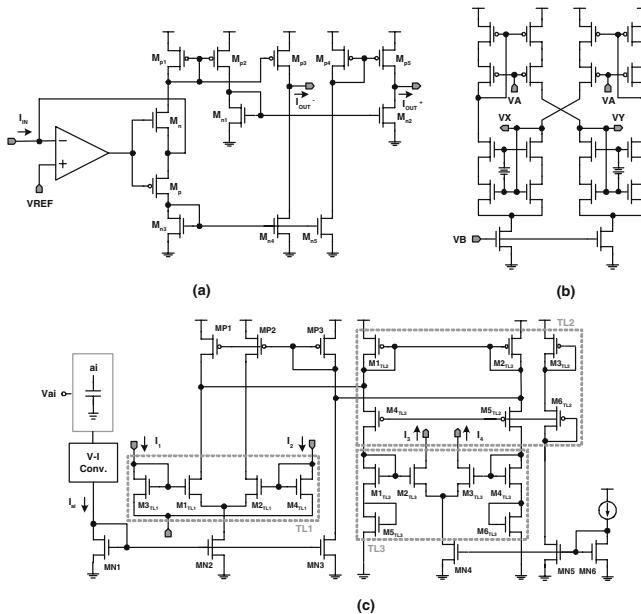


Fig. 5. The sub-circuits in the stochastic neuron. (a) Current conveyor with N:1 normaliser (b) Floating resistor (c) Sigmoid circuit.

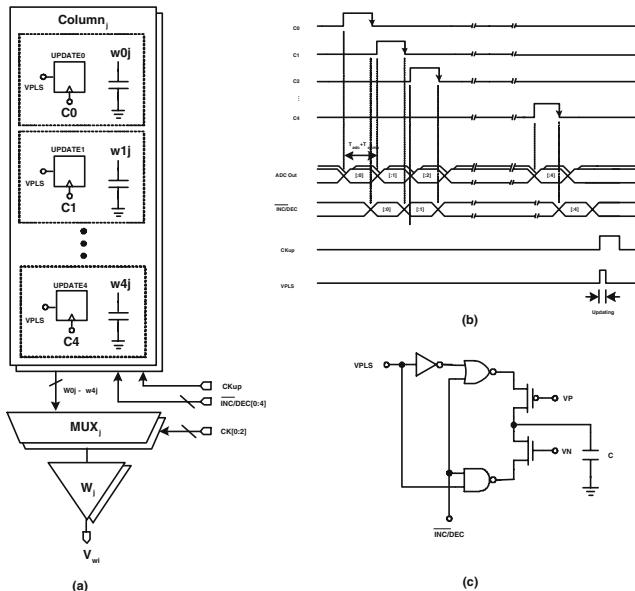


Fig. 6. (a) A multiplexing architecture for programming parameters in the CRBM system. (b) Clock signals for programming (c) Local updating circuit.

4 Circuit Design

The full system has been designed and fabricated with TSMC 0.35 μm technology. Fig.3 shows the layout of the full system. the layout of the full system. The circuit area excluding the pads is 3100 $\mu m \times$ 2800 μm , and the power consumption is 6.8mA. With a supply voltage of 3V, the parameter mapping is defined as Table I. The synapse module simply employs the "modified Chible multiplier" proposed in [7] as four-quadrant multipliers to calculate $w_{ij} \cdot s_j$. On the other hand, the noise generator proposed in [12] is implemented to generate 10 channels of uncorrelated noise on-chip. The following subsections describe the circuits of the neuron module and the programmable parameter array. In addition, Fig.4 shows the block diagram of the neuron indicating how hardware is implemented according to the mapping table (Table I), and the details of the circuit design are described below.

4.1 The Neuron

Fig.5 shows the sub-circuits in the neuron module. The current conveyer with N:1 normalisation (Fig.5(a)) uses an operational amplifier (OPamp) with negative feedback to provide a low-impedance input point for all synapses connected to the neuron [10]. The noise voltage from the noise generator is also transformed into a noise current by a differential pair and then connected to the same node. By controlling the tail current of the differential pair, the noise current can be scaled as σ in Eq.(1) [7]. The current I_{IN} proportional to $(\sum_j w_{ij} \cdot s_j + N_i(0, 1))$ in Eq.(1) is then normalised by N times through current mirrors, and subsequently transformed into a differential outputs, I_{OUT}^+ and I_{OUT}^- . With $I_1 = I_{OUT}^+$ and $I_2 = I_{OUT}^-$, the sigmoid circuit in Fig.5(c) employs three translinear loops (TL1-TL3) to generate a differential output ($I_3 - I_4$) whose value is approximately the sigmoidal function of $(I_1 - I_2)$ [13]. Moreover, the voltage representing a_i is converted into the current I_{ai} via the voltage-to-current converter (V-I converter) proposed in [9], controlling the slope of the sigmoid function. Finally, the current ($I_3 - I_4$) is converted into a voltage representing s_i by an Op-amp with a negatively feedback resistor. The feedback resistor is implemented by the circuit shown in Fig.5(b) to achieve a resistance of more than several $M\Omega$. with a compact area [11].

4.2 Programmable Parameter Array

The full microsystem contains 35 parameters (25 w_{ij} 10 a_i), which are arranged into a 5x7 array and multiplexed by the architecture shown in Fig.6(a). Fig.6(b) shows the corresponding digital-control signals. With C0-C4 decoded from CK[0:2], five parameter values in the same column ($w_{0j} - w_{4j}$) are selected sequentially by MUX_j and connected to the off-chip ADC. In refreshing mode, and connected to the off-chip ADC. In refresh mode, w_{0j} is first selected and compared with its target value as C0=1. The signal (INC/DEC[:0]) representing update direction is then determined and stored in a register next to the updating circuit. With the same procedure, sequential activation of C1-C4 determines and stores the update direction for the other four parameters. Once the update directions of all parameters are obtained, CKup=1 triggers updating circuit

to tune all parameter once. The updating step is controlled by the pulse-width of the signal VPLS. In training mode, update directions are multiplexed and registered into the parameter array in a similar manner, except for that the update directions are calculated from the MCD algorithm.

5 Measurement Results

Fig. 7(a) shows the measured output current of one multiplier in the synapse w40. With the output O of the neuron V4 sweeping from 1V to 2V, the output currents at IH, in response to different levels of w40, were measured. Obviously, the multiplier allowed the parameter w40 to have a rail-to-rail dynamic range and exhibited satisfactory linearity. Furthermore, the arithmetical zeros located at 1.5V precisely, agreeing with the mapping in Table I.

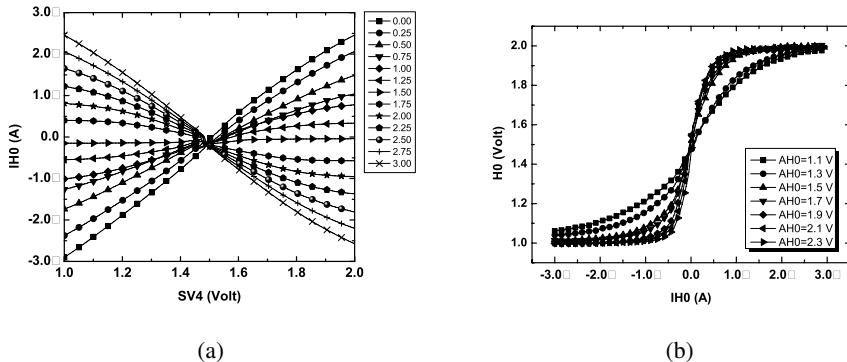


Fig. 7. The measured DC characteristics of (a) a four-quadrant multiplier and (b) a sigmoid circuit with its slope controlled by the voltage AH0

Fig. 7(b) shows the measured characteristics of the sigmoid circuit in the neuron H0. With the current at the input I (called IH0) sweeping from $-3\mu A$ to $3\mu A$, the voltage at the output O (called H0) was measured. Different curves correspond to different levels of the voltage AH0, which controls the slope of the sigmoid function. As $\varphi(1) = 0.462$ and the unit values of IH0 and H0 are 1 A and 0.5V, respectively, the adaptable range of AH0 corresponds to an adaptable range of [0.5, 2.5] for the parameter a_i , covering the required range ([0.5, 5]) set in Table I.

Fig. 8(a) shows the noise voltage (the top trace) measured at one channel of the noise generator. The signal fell in [1, 2](V), corresponding to a numerical range of [-1, 1] in software simulation. As the noise signal was sent into the neuron H0 with $w40=3V$ and $AH0=1.9V$, the measured neuron output, in response to all connected synapses having their VO sweeping between 1V and 2V, are shown in Fig. 8(b). The neuron output (the upper trace) swept the sigmoidal curve periodically with VO (the lower trace), and the noise input perturbed the curve significantly. The continuous-valued stochastic behaviour of the neuron in accordance with Eq. (II) was clearly demonstrated.

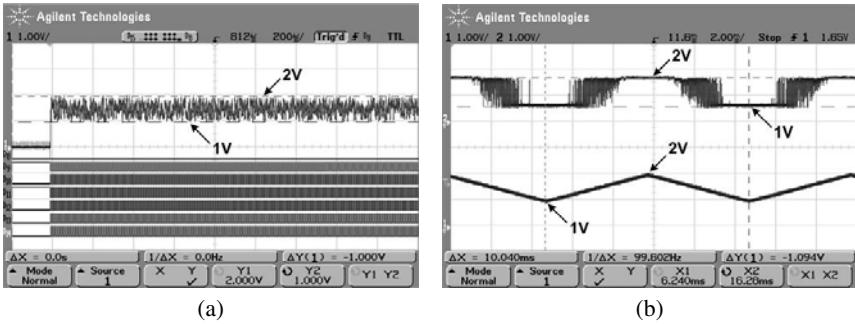


Fig. 8. The measured output of (a) one channel of noise voltage and (b) a stochastic neuron (upper trace) when its synapses had VO sweeping between 1.0 and 2.0 V (lower trace)

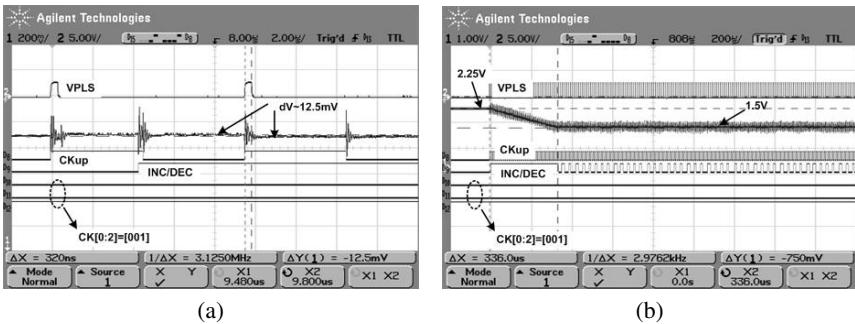


Fig. 9. (a) The measured updating stepsize of 12mV with VP=2.46V, VN=0.57V, and a pulse width of 320ns. (b) The measured programming process of a parameter which was adapted from 2.25V towards 1.5V.

Fig. 9(a) further shows the measured characteristic of the updating circuit in refreshing mode. With VP=2.46V, VN=0.57V, and a pulse width of 320ns for VPLS, an updating step of only 12mV was easily achieved for both incremental (INC/DEC=0) and decremental (INC/DEC=1) updates. The updating step could be further decreased by simply reducing the pulse width of VPLS, while the background noise of the oscilloscope and the switching noise made the updating step hardly visible. The programmability of parameter arrays was further tested by initialising a parameter to 2.25V and then adapting it towards 1.5V. Fig. 9(b) shows the measured parameter voltage (the second trace from top) and corresponding digital control signals. The parameter value adapted from 2.25V to 1.5V within 336 sec. As soon as the target value was achieved, the directional signal (INC/DEC) started to alternate between 1 and 0, refreshing the parameter at 1.5 reliably.

6 Conclusion

The VLSI circuits realising a scalable and programmable CRBM system have been designed, fabricated and tested. The preliminary measurement results demonstrate

satisfactory functionality of the synapse module, the neuron module, and their programmable parameters. By interconnecting multiple chips, the capability of the system to model high-dimensional biomedical data, as well as the feasibility of using noise-induced stochastic behaviour to enhance the robustness of analogue computation will be further examined and discussed.

Acknowledgement

The authors would like to acknowledge the TSMC and CIC for fabrication of the chip, and the National Science Council (NSC) in Taiwan for funding this project (Grant code: NSC 95-2221-E-007-115).

References

1. Tong, B.T., Johannessen, E.A., Lei, W., Astaras, A., Ahmadian, M., Murray, A.F., Cooper, J.M., Beaumont, S.P., Flynn, B.W., Cumming, D.R.S.: Toward a miniature wireless integrated multisensor microsystem for industrial and biomedical applications. *IEEE Sensors J.* 2(6), 628–635 (2002)
2. Johannessen, E.A., Wang, L., Wyse, C., Cumming, D.R.S., Cooper, J.M.A.: Biocompatibility of a Lab-on-a-Pill Sensor in Artificial Gastrointestinal Environments. *IEEE Transactions on Biomedical Engineering* 53(11), 2333–2340 (2006)
3. Genov, R., Cauwenberghs, G.: Kerneltron: support vector machine in silicon. *IEEE Trans. on Neural Networks* 14(8), 1426–1433 (2003)
4. Hsu, D., Bridges, S., Figueroa, M., Diorio, C.: Adaptive Quantization and Density Estimation in Silicon. In: *Advances in Neural Information Processing Systems*, pp. 1083–1090. MIT Press, Cambridge (2002)
5. Chen, H., Murray, A.F.: Continuous restricted Boltzmann machine with an implementable training algorithm. *IEE Proceedings-Vision Image and Signal Processing* 150(3), 153–158 (2003)
6. Chen, H., Fleury, P.C.D., Murray, A.F.: Continuous-valued probabilistic behavior in a VLSI generative model. *IEEE Trans. on Neural Networks* 17(3), 755–770 (2006)
7. Chen, H., Fleury, P., Murray, A.F.: Minimizing Contrastive Divergence in Noisy, Mixed-mode VLSI Neurons. In: *Advances in Neural Information Processing Systems (NIPS 2003)*. MIT Press, Camridge (2004)
8. Lu, C.C., Hong, C.Y., Chen, H.: A Scalable and Programmable Architecture for the Continuous Restricted Boltzmann Machine in VLSI. In: *IEEE International Symposium on Circuits and Systems* (2007)
9. Cauwenberghs, G.: An Analog VLSI Recurrent Neural Network Learning a Continuous-Time Trajectory. *IEEE Transactions on Neural Networks* 7(2), 346–361 (2003)
10. Liu, S.-C., Kramer, J., Indiveri, G., Delbrück, T., Douglas, R.: *Analog VLSI: Circuits and Principles*. MIT Press, MA (2002)
11. Al-Sarawi, S.F.: A novel linear resistor utilizing MOS transistors with identical sizes and one controlling voltage. *Microelectronics Journal* 33(12), 1059–1069 (2002)
12. Cauwenberghs, G.: Delta-sigma cellular automata for analog VLSI random vector generation. *IEEE Transaction on Circuit and System II: Analog and Digital Signal Processing* 46(3), 240–250 (1999)
13. Diotalevi, F., Valle, M., Bo, G.M., Biglieri, E., Caviglia, D.D.: Analog CMOS current mode neural primitives. *Circuit and System II: Analog and Digital Signal Processing*. In: *IEEE International Symposium on Circuits and Systems* (2000)

Minimising Contrastive Divergence with Dynamic Current Mirrors

Chih-Cheng Lu and H. Chen

The Dept. of Electrical Engineering,
The National Tsing-Hua University, Hsin-Chu, Taiwan 30013
hchen@ee.nthu.edu.tw

Abstract. Implementing probabilistic models in Very-Large-Scale-Integration (VLSI) has been attractive to implantable biomedical devices for improving sensor fusion. However, hardware non-idealities can introduce training errors, hindering optimal modelling through on-chip adaptation. This paper investigates the feasibility of using the dynamic current mirrors to implement a simple and precise training circuit. The precision required for training the Continuous Restricted Boltzmann Machine (CRBM) is first identified. A training circuit based on accumulators formed by dynamic current mirrors is then proposed. By measuring the accumulators in VLSI, the feasibility of training the CRBM on chip according to its minimizing-contrastive-divergence rule is concluded.

Keywords: Minimising Contrastive Divergence, Dynamic Current Mirrors, Probabilistic Model, Boltzmann Machine, On-chip training.

1 Introduction

As probabilistic models are able to generalise the natural variability in data, the VLSI implementation of probabilistic models has been attractive to implantable biomedical devices [1] [2]. However, seldom probabilistic models are amenable to the VLSI implementation. Among the proposed probabilistic models in VLSI [3] [4] [5], the Continuous Restricted Boltzmann Machine (CRBM) has been shown capable of modelling biomedical data with a hardware-friendly training algorithm, which minimises the contrastive divergence (MCD) between training and modelled distributions [6] [7]. However, experiments in [7] revealed that offsets in training circuits limited the minimum achievable divergence, preventing the CRBM microsystem from modelling data optimally. To make useful the VLSI implementation of the CRBM and many other models, it is important to develop a simple circuit capable of realising contrastive training rules with satisfactory precision.

This paper examines the feasibility of using dynamic current mirrors to realise contrastive-divergence training algorithms on-chip with satisfactory precision. As continuous-valued models are inherently more sensitive to the existence of training errors, the satisfactory precision refers to the capability of training the CRBM microsystem to model both artificial and real biomedical (ECG) data satisfactorily.

2 The CRBM Model

The CRBM consists of one visible and one hidden layers of stochastic neurons with inter-layer connections only [7]. The number of visible neurons corresponds to the dimension of data, while that of hidden neurons is chosen according to data complexity [7]. Let w_{ij} represent the bi-directional connection between neurons s_i and s_j . The stochastic state of a neuron s_i is defined by [7]

$$s_i = \varphi_i(a_i \cdot (\sum_j w_{ij} \cdot s_j + N_i(0, \sigma))) \quad (1)$$

where $N_i(0, \sigma)$ represents a Gaussian noise with zero mean and variance σ^2 , and $\varphi_i(\cdot)$ a sigmoid function (e.g. $\tanh(\cdot)$) with asymptotes at ± 1 . Parameter a_i controls the slope of the sigmoid function and thus the variance of s_i , such that the neuron is either near-deterministic (small a_i), or continuous-stochastic (moderate a_i), or binary-stochastic (large a_i). Let λ represent the parameter $\{w_{ij}\}$ or $\{a_i\}$. Parameters in a CRBM microsystem are trained by the simplified MCD algorithm [3]

$$\Delta\lambda = \eta_\lambda \cdot (\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4) \quad (2)$$

where \hat{s}_i and \hat{s}_j denotes the one-step Gibbs-sampled states [7], η_λ the updating rate, and $\langle \cdot \rangle_4$ taking the expectation over four training data. The difference between $\langle s_i \cdot s_j \rangle_4$ and $\langle \hat{s}_i \cdot \hat{s}_j \rangle_4$ corresponds to the contrastive divergence between training and modelled distributions [6] and has to be minimised. For training $\{a_i\}$, s_j and \hat{s}_j in Eq.(2) are replaced by s_i and \hat{s}_i , respectively.

3 Maximum Offsets Tolerable by the CRBM

The CRBM has been realised as a VLSI microsystem containing six neurons with on-chip training circuits [3]. However, hardware nonidealities in training circuits prevents the CRBM system from modelling data optimally, and it was shown that the overall effect of hardware nonidealities can be modelled as the "biased" training algorithm

$$\Delta\lambda = \eta_\lambda \cdot (\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4 + \Delta_T) \quad (3)$$

where Δ_T represents the offset that limits the minimum contrastive divergence achievable by on-chip training circuits. Although the offset varies from one circuit to another, it is assumed to be identical in simulation for simplicity. Based on the software-hardware mapping derived in [3], the following subsections simulate the behaviour of a CRBM microsystem with Eq.(3), and identify the maximum offsets (Δ_T) the system can tolerate. The value of tolerable offsets will be given in terms of percentage, normalized with respect to the maximum value of $|\langle s_i \cdot s_j \rangle_4| = 1$. (i.e. $\Delta_T = 1\%$ refers to $\Delta_T = 0.01$ in Eq.(3)).

3.1 Quantitative Index for Offset Tolerance

As a *generative* model, the CRBM learns to "regenerate" training data distribution. To identify an index for measuring quantitatively how well the CRBM models a dataset, the

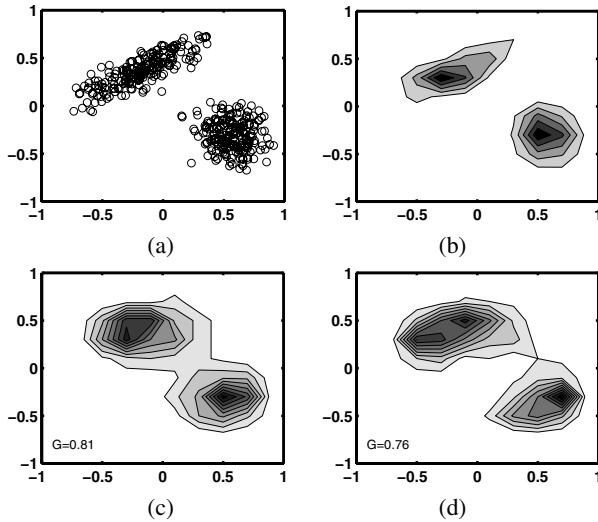


Fig. 1. (a) Artificial training data with two clusters of Gaussian-distributed data points. (b) The statistical density of the training data. (c)(d): The statistical density of 20-step reconstructions generated by the CRBM after (c)20000 (d)30000 training epochs.

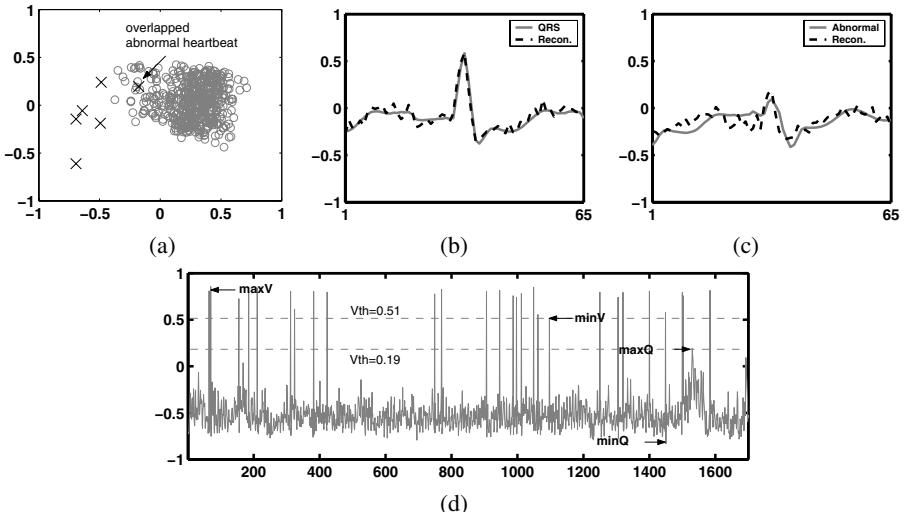


Fig. 2. (a) The projection of 500 ECG training data to its first two principle components. The projection of the five abnormal ECGs are denoted by black crosses. (b)(c)(d): Results of training the CRBM to model ECG data with $\Delta_T = 0.2\%$ for all parameters. (b) (b) The normal and (c)the abnormal ECGs in training dataset (grey) and the reconstruction by the trained CRBM (dashed). (d) Responses of hidden neuron h_3 to 1700 testing data. maxV , minV , maxQ , and minQ correspond to the maximum and minimum responses to abnormal heartbeats, and maximum and minimum responses to normal heartbeats, respectively.

Table 1. Tolerable offsets with four-data, sign-valued training algorithm, four-data, real-valued training algorithm, and single-datum training algorithm

METHOD\MODELLED DATA	TWO-CLUSTER	ECG
Four-data, sign-valued	1%	0.2%
Four-data, real-valued	1%	0.3%
Single-data, real-valued	0.2%	0.05%

CRBM with two visible and four hidden neurons, as shown in Fig 1 was first trained with the ideal algorithm (Eq.(2)) to model the artificial data in Fig 1(a). The dataset contains one elliptic and one circular clusters of 200 Gaussian-distributed data points whose statistical density is shown in Fig 1(b). Let $P^T(\mathbf{v})$ and $P^M(\mathbf{v})$ represent the distribution of training data and the distribution modeled by the CRBM, respectively. The *Kulback-Leibler (KL) Divergence* defined as Eq.(4) [8] measures the difference between $P^T(\mathbf{v})$ and $P^M(\mathbf{v})$.

$$G = \sum_{\mathbf{v}} P^T(\mathbf{v}) \log \frac{P^T(\mathbf{v})}{P^M(\mathbf{v})} \quad (4)$$

where \mathbf{v} denotes the subset of visible states, and G equals zero when $P^T(\mathbf{v}) = P^M(\mathbf{v})$. As not all distributions can be described by explicit equations, $P^T(\mathbf{v})$ and $P^M(\mathbf{v})$ were statistically-estimated by dividing the two-dimensional space into 10x10 square grids, counting the number of data points in each grid, and normalising the counts with respect to the total number of data points. Fig 1(c)(d) shows the statistical density of 20-step reconstructions generated by the CRBM after 20000 and 30000 training epochs, respectively. The G values calculated according to Eq.(4) are shown at the bottom-left corner of each subfigure, indicating that the KL-divergence is a reliable index for measuring quantitatively the similarity between training and modelled distributions. Similar results are obtained for other data like doughnut-shaped distribution. As the training updates of most parameters become negligible after $G < 0.8$, it is chosen as the criterion for identifying the tolerable offsets for the CRBM. When all parameters ($\{w_{ij}\}$, $\{a_{vi}\}$, and $\{a_{hi}\}$) experience offsets, the maximum offsets the CRBM can tolerate to model artificial data was identified to be 1% (Table 1).

3.2 Modelling Real Heartbeat Data with Offsets

The tolerable offset for modelling high-dimensional, real-world data was examined in the context of recognising electrocardiograms (ECG), extracted from the MIT-BIH database as in [9] [10]. The training dataset contains 500 heartbeats with only 5 abnormal heartbeats. The testing dataset contains 1700 heartbeats with 27 abnormal heartbeats. Each ECG trace was sampled as a 65-dimensional datum, and Fig 2(a) shows the projection of the training dataset onto its first two principle components. Although the dimension reduction made the quantitative index G remain applicable, pilot simulation showed that modelling training data satisfactorily did not guarantee the detection of abnormal heartbeats with 100% accuracy. This was because the projected distributions of normal and abnormal heartbeats at low dimension overlap with each other, as obviated

by Fig 2(a). (the data remain separable in high dimensions). Therefore, detection with 100% accuracy was used as a stricter criterion for identifying the tolerable offsets for modelling ECG data. Extensive simulations further showed that the CRBM tolerated an offset of only 0.2% to model ECG data (Table I). With $\Delta_T = 0.2\%$, the trained CRBM was able to reconstruct both normal and abnormal ECG signal satisfactorily, as shown in Fig 2(b)(c). In addition, Fig 2(d) shows the responses of hidden neuron h_3 to 1700 testing data $\{\mathbf{d}\}$, calculated according to Eq.(5). The abnormal heartbeats can be detected with 100% accuracy by setting any threshold between minV and maxQ.

$$h_3 = \varphi(a_3 \cdot (\mathbf{w}^{(3)} \cdot \mathbf{d})) \quad (5)$$

3.3 Tolerable Offsets for Different Training Strategies

As implementing training circuits with an offset less than 0.2% is quite a challenge, we further investigated the possibility of releasing the strict requirement with two modified training strategies, (a) updating parameters with real-valued contrastive divergence ($\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$) instead of taking only its sign and (b) updating parameters with real-valued contrastive divergence datum by datum [1]. The last two rows of Table I summarise the maximum offsets the CRBM can tolerate to model artificial and ECG data with the different training strategies. Comparison with the first row indicates that four-data, real-valued adaptation does enhance the tolerance against offsets slightly, while single-datum adaptation degrades the tolerance significantly. The latter demonstrates that calculating the expectation value, i.e. accumulating opinions from multiple data, is important for estimating the "contrastive divergence" between distributions.

4 The Contrastive-Divergence Training Circuit Based on Dynamic Current Mirrors

Although an offset smaller than 0.3% remains challenging, the dynamic current mirrors (DCMs) were reported to have errors smaller than 500ppm [12] [13]. Therefore, we propose the training circuit in Fig.3 that uses DCMs to calculate $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$ or $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$ in contrastive-divergence training rules. Each DCM works as a register, using the same transistor (M1 or M2 in Fig.3(a)) to sample and transfer currents. The mismatching errors in conventional current mirrors, i.e. the main cause of training offsets in [10], are thus avoided. Fig.3(a) shows the accumulator consisting of one NMOS (M1-M1c) and one PMOS (M2-M2c) DCMs. The DCM training circuit realises the single-datum training algorithm by simply three steps. I_{in} in I_{in} in Fig.3(a) represents $s_i \cdot s_j$ or $\hat{s}_i \cdot \hat{s}_j$ calculated by the multiplier. At the first step with switches S_{IN} , S_{D1} , and S_{G1} closed, the current representing $s_i \cdot s_j$ is sampled into the NMOS DCM, and then stored as the voltage across the capacitor $C1$ after S_{IN} and S_{G1} become opened. At the second step, switches S_{D2} and S_{G2} are closed to copy the same current into the PMOS DCM. The sampled current is stored as a voltage across the capacitor $C2$ after S_{D2} and S_{G2} are opened. At the third step, I_{in} representing $\hat{s}_i \cdot \hat{s}_j$ is sampled and stored in the NMOS DCM by repeating the first step. Finally, closing S_{D1} , S_{D2} , and S_{OUT} gives an output current proportional to $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$. To implement the four-data

training algorithm, the circuit in Fig.3(a) functions as an accumulator that calculates $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$. The first and the second steps described above are carried out once to store $s_i \cdot s_j$ of the first datum into the PMOS DCM. At the third step, let I_{in} correspond to $s_i \cdot s_j$ of the second datum. Closing S_{IN} , S_{D1} , S_{G1} , and S_{D2} stores the sum of $s_i \cdot s_j$ of both data into the NMOS DCM. Repeating the second and the third steps alternatively then sums up $s_i \cdot s_j$ of multiple data and stores it into the NMOS DCM. To calculate the contrastive divergence, $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$ two identical accumulators are employed as in Fig.4(b). The top accumulator calculates $\langle s_i \cdot s_j \rangle_4$ and stores the value into its PMOS DCM, while the bottom one simply stores $\langle \hat{s}_i \cdot \hat{s}_j \rangle_4$ into NMOS DCM. As soon as switch S_{OUT} is closed, an output current proportional to $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$ is produced. Finally, I_{OUT} is directed into the charge amplifier in Fig.3(c) which functions as the low-impedance reference voltage (V_{REF}) in Fig.3(b). I_{OUT} then modifies the voltage stored across C_F , which represents a parameter value of the CRBM.

The DCM training circuit in Fig.3 is capable of realising the three contrastive-divergence training algorithms in Table 1 as well as other contrastive training rules in [8] [14] [15] [16]. The learning rate η_λ in Eq.(2) can be defined by the period of closing S_{OUT} . Unlike the DCMs in [12] [13], the DCMs in Fig.3 have not only to transport currents of various values but also to function as both accumulators and subtractors, coping with a wide range of currents. Cascode transistors M1a and M2a in Fig.3(a) are therefore employed to reduce the effect of channel-length modulation by fixing the drain voltage of M1 and M2, respectively.

The DCM training circuit also suffers from offsets introduced by the nonlinearity of multipliers and the charge-injection errors. The former can be easily avoided by using a multiplier with symmetric outputs, for example, the modified Chible multiplier proposed in [3]. As for the latter, complementary transistors can be used as switches to compensate for charge-injection errors. However, the simulation normally underestimates the charge-injection errors. The DCM accumulators in Fig.3(a) and (b), excluding the multiplier, were thus fabricated with the TSMC 0.35um 2P4M CMOS process to investigate the precision achievable by the proposed training circuits.

For the single-datum training algorithm, I_{in} was designed to range from $1\mu A$ to $3\mu A$ corresponding to $s_i \cdot s_j = -1$ and $\hat{s}_i \cdot \hat{s}_j = 1$, respectively. To minimise the dependence of charge injection on I_{in} , the charge-injection error is minimised at $I_{in} = 2\mu A$. For four-data training algorithm, I_{in} was designed to range from $0.25\mu A$ to $0.75\mu A$, such that the accumulation of four data still ranges from $1\mu A$ to $3\mu A$, allowing the minimisation of charge-injection error to remain at $I_{in} = 2\mu A$.

5 Measurement Results

With I_{in} generated from a Source Meter (Keithley 2602), the output of the DCM accumulators were connected to a current-voltage(I-V) converter, which emulated V_{REF} in Fig.3 and converted I_{out} into a voltage of $V_o = V_{REF} - H_f \cdot I_{out}$ with $H_f = 1650(V/A)$. The voltage change $\Delta V_o = V_o - V_{REF} = -H_f \cdot I_{out}$ at the instant of closing S_{OUT} was then measured. With digital-control clocks generated by a Field-Programmable-Gate-Array (FPGA) chip, the DCM accumulators were set easily to calculate $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$ or $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$.

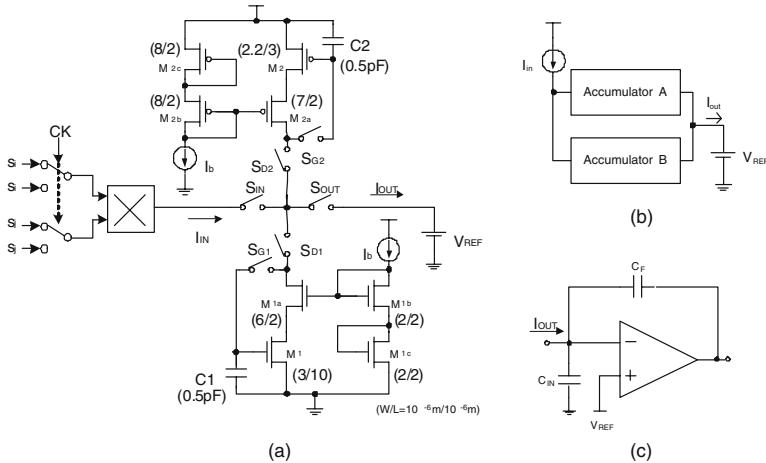


Fig. 3. The proposed DCM training circuit consisting of (a) a multiplier and the DCM accumulator calculating $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$, or (b) the DCM accumulator calculating $(\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4)$, and, (c) the charge amplifier

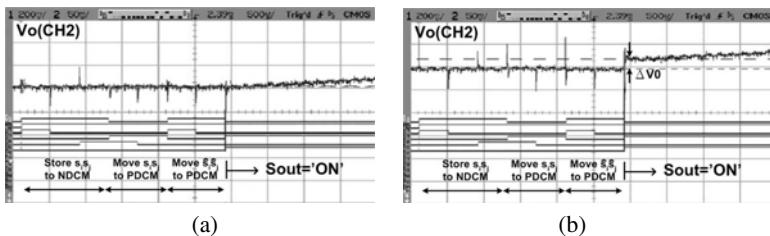


Fig. 4. Measured $V_0 = V_{REF} - I_{out} \cdot H_f$ for single-datum training with $I_{in} = 1\mu A$. (a) without offset (b) with offset. The digital signals from top to bottom correspond to S_{IN} , S_{D1} , S_{G1} , S_{D2} , S_{G2} , and S_{OUT} in Fig. 3

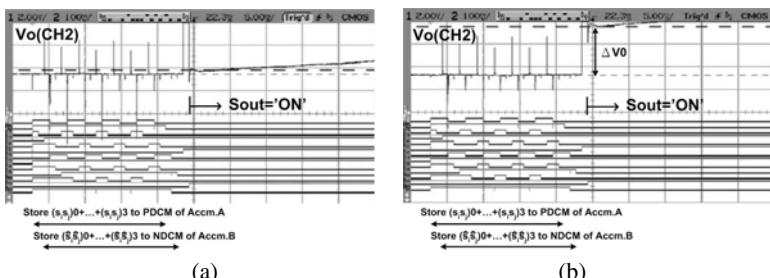


Fig. 5. Measured $V_0 = V_{REF} - I_{out} \cdot H_f$ for four-data training with $I_{in} = 0.25\mu A$. (a) without offset (b) with offset. The digital signals from top to bottom correspond to S_{IN} , S_{D1} , S_{G1} , S_{D2} , S_{G2} , and S_{OUT} for accumulator A and those of accumulator B in Fig. 3

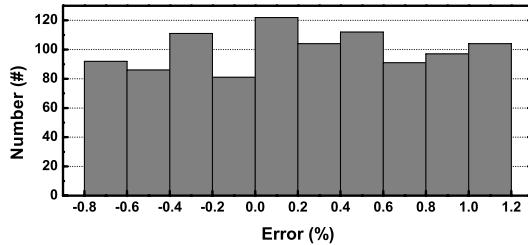


Fig. 6. Statistical histogram of the offset errors measured from a DCM training circuit set to carry out single-data training algorithm

Table 2. Measured offsets in the calculation of $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$

$s_i \cdot s_j$	$\hat{s}_i \cdot \hat{s}_j$	MEAN ERROR	STD. DEV.
$1\mu A$	$1\mu A$	0.38%	0.56%
$2\mu A$	$2\mu A$	0.19%	0.39%
$3\mu A$	$3\mu A$	0.45%	0.63%

Table 3. Measured offsets in the calculation of $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$

$s_i \cdot s_j (\mu A)$	$\hat{s}_i \cdot \hat{s}_j (\mu A)$	MEAN ERROR	STD. DEV.
$0.25+0.25+0.25+0.25$	$0.25+0.25+0.25+0.25$	1.31%	0.38%
$0.5+0.5+0.5+0.5$	$0.5+0.5+0.5+0.5$	2.21%	0.55%
$0.75+0.75+0.75+0.75$	$0.75+0.75+0.75+0.75$	3.61%	0.63%

For calculating $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$, the measured V_o in response to a constant I_{in} of $1\mu A$, i.e. $s_i \cdot s_j = \hat{s}_i \cdot \hat{s}_j = -1$, is shown in Fig. 4. Although ΔV_0 ideally equaled zero, ΔV_0 measured from the same circuit either approximated zero or varied from one trial to another. Similar results were observed when calculating $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$, as shown in Fig. 5. Normalising ΔV_0 with respect to $1.65V$ (the ΔV_0 for $I_{out} = 1\mu A$ representing $|S_i \cdot S_j| = 1$) gives the offset errors in terms of percentage. Fig. 6 shows the statistical distribution of the offsets measured from 1000 trials of calculating $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$. Interestingly, the offsets exhibit a uniform distribution instead of staying constant, and the variance is greater than the mean value.

The offsets in the DCM accumulators were caused by charge-injection errors, leakage currents of C1 and C2, and clock jitters generated by the FPGA chip. To investigate the contribution of leakage currents, two types of digital clocks were used to buffer I_{in} to I_{out} by storing I_{in} in the NMOS DCM, transferring it to the PMOS DCM, and subsequently outputting the current. One clock differs from the other mainly by shortening the period of opening S_{G1} and S_{G2} . Shortening the period improved the mean errors from -8.09% to -6.38%, while the standard deviations of the two cases are comparable

(3.45%). Therefore, leakage currents mainly affect the mean errors, while clock jitters have dominant effects on the variance. Accumulating four I_{in} caused mean errors to increase by more than four times, while the standard deviations remained about the same. Charge injection thus also affected mainly the mean errors.

Table 2 summarises the performance of the DCM accumulator in calculating $(s_i \cdot s_j - \hat{s}_i \cdot \hat{s}_j)$ with different I_{in} . The mean errors became significantly smaller than 6.38%, indicating that charge-injection and leakage-current errors in the NMOS and PMOS DCMs cancelled with each other largely through the subtraction operation. Complete cancellation was difficult because the current representing $(s_i \cdot s_j)$ in the PMOS DCM unavoidably suffered from extra switching events than the current representing $(\hat{s}_i \cdot \hat{s}_j)$ in the NMOS DCM. Moreover, Table 2 reveals that the randomness in offsets was also reduced by the subtraction operation. Table 3 summarises the performance in calculating $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$. Compared to Table 2, the mean errors all became higher because charge-injection and leakage-current errors were accumulated. Nevertheless, the standard deviations in both tables are comparable, confirming that clock jitters dominate to introduce the randomness in offsets.

6 Modelling Data with Uniformly-Distributed Offsets

To simulate the performance of a CRBM microsystem with the proposed DCM training circuits, Δ_T in the training rule was replaced by a uniform random variable with nonzero mean (μ_T) and a standard deviation (σ) of 0.7%, the measured maximum deviation. Under the existence of Δ_T for all parameters, the maximum mean offsets (μ_T) that the CRBM can tolerate to model both artificial and ECG data are identified and summarised in Table 4. Compared to Table 1, the tolerance is slightly improved. This feature agrees with the finding that randomness releases the precision required for training a multi-layer-perceptron [17] [18]. The CRBM is able to correct training errors whenever the random offset is small, and thus to discourage the saturation of parameter values. Therefore, it is important to know when to stop training once the data distribution is modelled, so as to prevent Δ_T from dominating to causes all parameters to saturate. Fortunately, the G value could be used as a reliable indicator for when to stop.

Table 4. Tolerable mean offsets by the CRBM with different training strategies

METHOD\MODELLED DATA	TWO-CLUSTER	ECG
Four-data, sign-valued	2%	0.3%
Four-data, real-valued	2%	0.5%
Single-data, real-valued	0.5%	0.1%

7 Conclusion

The feasibility of minimising contrastive divergence on-chip with DCMs has been carefully investigated by both behavioural simulation of the CRBM microsystem and the

VLSI implementation of DCM accumulators. The simulation indicates that the CRBM can tolerate a maximum offset of only 0.3% to model real biomedical (ECG) data satisfactorily, and that the tolerance can be slightly-improved by real-valued adaptation. On the other hand, measurement results of DCM accumulators indicate that the accumulation errors in DCMs can be largely cancelled by the subtraction operation essential for the contrastive-divergence training. As the mean offsets in Table II are all smaller than 0.5%, i.e. the tolerable offset for four-data, real-valued training in Table 4, using four DCM accumulators (Fig. 3(a)) to calculate $\langle s_i \cdot s_j \rangle_4 - \langle \hat{s}_i \cdot \hat{s}_j \rangle_4$ would allow us to avoid the accumulation error in Table 3 while achieving satisfactory precision. This suggestion will be further confirmed with the VLSI implementation of the full training circuit.

Acknowledgement

The authors would like to acknowledge the TSMC and CIC for fabrication of the chip, and the National Science Council (NSC) in Taiwan for funding this project (Grant code: NSC 95-2221-E-007-115).

References

1. Schwartz, A.B.: Cortical Neural Prosthetics. *Annual Review Neuroscience*, 487–507 (2004)
2. Lebedev, M.A., Nicolelis, M.A.L.: Brain-machine interfaces: past, present and future. *TRENDS in Neuroscience* 29[9], 536–546 (2006)
3. Chen, H., Fleury, P., Murray: Continuous-Valued Probabilistic Behaviour in a VLSI Generative Model. *IEEE Trans. on Neural Networks* 17(3), 755–770 (2006)
4. Genov, R., Cauwenberghs, G.: Kerneltron: support vector machine in silicon. *IEEE Trans. on Neural Networks* 14(8), 1426–1433 (2003)
5. Hsu, D., Bridges, S., Figueiroa, M., Diorio, C.: Adaptive Quantization and Density Estimation in Silicon. In: *Advances in Neural Information Processing Systems* (2002)
6. Hinton, G.E.: Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation* 14(8), 1771–1800 (2002)
7. Chen, H., Murray, A.F.: A Continuous Restricted Boltzmann Machine with an Implementable Training Algorithm. *IEE Proc. of Vision, Image and Signal Processing* 150(3), 153–158 (2003)
8. Hinton, G.E., Sejnowski, T.J.: Learning and Relearning in Boltzmann Machine. In: Rumelhart, D., McClelland, J.L., The PDP Research Group (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pp. 283–317. MIT, Cambridge (1986)
9. MIT-BIH Database Distribution, <http://ecg.mit.edu/index.htm>
10. Chen, H., Fleury, P., Murray, A.F.: Minimizing Contrastive Divergence in Noisy, Mixed-mode VLSI Neurons. In: *Advances in Neural Information Processing Systems*, vol. 16 (2004)
11. Chiang, P.C., Chen, H.: Training Probabilistic VLSI models On-chip to Recognise Biomedical Signals under Hardware Nonidealities. In: *IEEE International Conf. of Engineering in Medicine and Biology Society* (2006)
12. Wegmann, G., Vittoz, E.: Analysis and Improvements of Accurate Dynamic Current Mirrors. *IEEE J. of Solid-State Circuits* 25[3], 699–706 (1990)
13. Fleury, P., Chen, H., Murray, A.F.: On-chip Contrastive Divergence Learning in Analogue VLSI. In: *Proc. of the International Joint Conference on Neural Networks* (2004)

14. Teh, Y.W., Hinton, G.E.: Rate-coded Restricted Boltzmann Machine for Face Recognition. In: Advances in Neural Information Processing System. MIT Press, Cambridge (2001)
15. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice Hall, Englewood Cliffs (1998)
16. Peterson, C., Anderson, J.R.: A Mean Field Theory Learning Algorithm for Neural Networks. *Complex Systems* 1, 995–1019 (1987)
17. Murray, A.F.: Analogue Noise-enhanced Learning in Neural Network Circuits. *Electronics Letters* 27(17), 1546–1548 (1991)
18. Murray, A.F., Edwards, P.J.: Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training. *IEEE Trans. on Neural Networks* 5(5), 792–802 (1994)

Spiking Neural Network Self-configuration for Temporal Pattern Recognition Analysis

Josep L. Rosselló, Ivan de Paúl, Vincent Canals, and Antoni Morro

Electronic Systems Group, Physics Department,
Universitat de les Illes Balears (UIB),
07122 Palma de Mallorca, Spain
j.rossello@uib.es

Abstract. In this work we provide design guidelines for the hardware implementation of Spiking Neural Networks. The proposed methodology is applied to temporal pattern recognition analysis. For this purpose the networks are trained using a simplified Genetic Algorithm. The proposed solution is applied to estimate the processing efficiency of Spiking Neural Networks.

Keywords: Neural Networks, Spiking Neural Networks, Hardware implementation of Genetic Algorithms.

1 Introduction

The development of efficient solutions for the hardware implementation of neural systems is currently one of the major challenges for science and technology. Due to their parallel-processing nature, among other applications, neural systems can be used for real-time pattern recognition tasks and to provide quick solutions for complex problems that are intractable using traditional digital processors [1,2]. The distributed information processing of Neural Networks also enhances fault tolerance and noise immunity with respect to traditional sequential processing machines. Despite of all these processing advantages, one of the main problems of dealing with neural systems is the achievement of optimum network configurations since network complexity increases exponentially with the total number of neural connections. Therefore, the development of learning strategies to quickly obtain optimum solutions when dealing with huge network configuration spaces is of high interest for the research community.

Recently, a lot of research has been focused on the development of Spiking Neural Networks (SNN) [3] as they are closely related to real biological systems. In SNN information is codified in the form of voltage pulses called Action Potentials (APs). At each neuron cell the AP inputs are weighted and integrated in a single variable defined as the Post-Synaptic-Potential (PSP). The PSP is time dependent and decays when no APs are received. When input spikes excite the PSP of a neuron sufficiently so that it is over a certain threshold, an Action Potential is emitted by the neuron and transmitted to the rest of the network.

In this work we present a practical implementation of SNN. We also develop a simple architecture that can be used for training SNN. The proposed self-learning solution has been applied for temporal pattern recognition. Based on this study, we also provide a new metric to estimate the processing capacity of NN. The rest of this paper is organized as follows: in section 2 we show the proposed SNN self-learning architecture, while in section 3 we apply the proposed system to temporal pattern recognition analysis. Finally in section 4 we present the conclusions.

2 Digital SNN Architecture

2.1 Digital Spiking Neuron Model

As mentioned in the previous section, in SNN the information is codified in the form of voltage pulses. We used a simplified digital implementation of the real behavior of biological neurons. In the proposed system, the PSP decay after each input spike is selected to be linear instead of the real non-linear variation while the refractory period present after each spike emission has been neglected. The main objective of this work is not to provide an exact copy of the real behavior of biological systems but to develop a useful Neural Network configuration technique. In Fig. 1 it is shown an example of the dynamic behavior of the digital model implemented.

In the digital version implemented the PSP is codified as a digital number. At each spike integration the PSP is increased a fixed value that depends on the type and the strength of the connection. Therefore, positive (negative) increment values are associated to excitatory (inhibitory) connections. Each neuron is implemented using a VHDL code in which the connection strength is selected

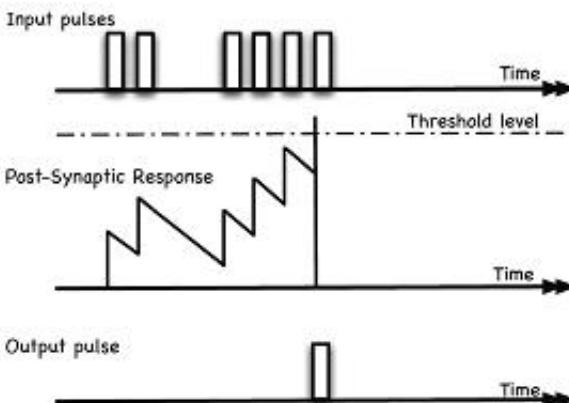


Fig. 1. Dynamic behavior of the digital implementation of spiking neurons. Action potentials are represented as digital pulses while post-psynaptic potential variation is assumed to be linear with time. The typical refractory period after each output spike has been neglected in this model.

to be a fraction of the neuron threshold (in particular, we selected a fraction of $\pm 2/5$ for both excitatory and inhibitory connections).

2.2 Self-learning Architecture

The proposed self-learning architecture is shown in Fig. 2. It consists in two basic blocks, a Genetic Algorithm Circuitry (GAC) and a Fitness Circuitry (FC). The GAC generates new configurations based on the better configuration obtained, that is stored in the configuration register. Using a Random Number Generator (RNG) a random mutation vector is generated. The mutation vector is operated using XOR gates with the better configuration found until the moment (placed in the configuration register). The result is a new configuration (binary output of XOR block) that is equal to the previous except in those cases where the RNG provides a HIGH state. The new configuration is directly applied to the SNN when the controlling signal of the GAC multiplexer (SL) is HIGH (self-learning selection). When signal SL is LOW (operation mode) the better configuration obtained until that moment is applied to the SNN.

The FC block evaluates the aptness of each new configuration for a selected network task. During the training mode an evaluation circuitry (EC) compares the SNN behavior with respect to the expected behavior, thus evaluating a cost function (the configuration fitness). The value obtained in this process is then compared to the one associated to the better configuration at the moment (stored at the fitness register). When a better fitness is found at the end of the evaluation time, the digital comparator output is set to a HIGH state and both the fitness and the configuration registers are updated with the new values. When the system is in operation mode, the SNN configuration is fixed to the better solution obtained at the moment. A global reset is used to start with pre-selected initial conditions.

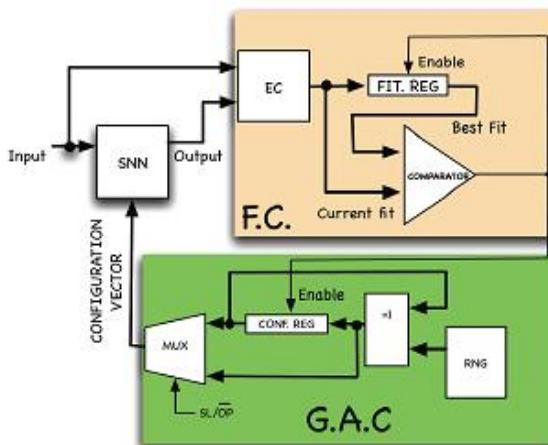


Fig. 2. Block structure for the dynamic configuration of SNN. The GAC block is used for the network configuration while the FC block evaluates the network efficiency.

2.3 Random Vector Generation

The mutation vector is used to generate a new configuration that is equal to the previous one, except in those cases where the RNG provides a HIGH state. The election of the RNG is important since all the possible mutation vectors must have the same probability of being generated. Therefore, the percentage of mutation ranges between the 0% (mutation vector 00...0) and the 100% (mutation vector 11...1). Using this strategy we ensure the possibility of moving from a local minima to a deeper (and therefore better) minima. Make note that, since the system is directly implemented in hardware it can sweep millions of different configuration vectors per second, thus obtaining a good solution in a reasonable time (although, of course, the absolute minimum is not guaranteed).

For the generation of the mutation vector we can choose either a pseudo-random or a random number generator. In FPGA applications we can use the first one since it is easily implemented using LFSR registers. For VLSI implementations we can choose a lower-cost solution as a true random number generator [4]. The solution proposed in [4] represent a lower cost in terms of hardware resources if compared to LFSR counters.

3 Application to Temporal Pattern Recognition

We applied the proposed SNN architecture to evaluate the processing behavior of various networks. The selected SNN task is the temporal pattern recognition (that is directly related to the “memory” capacity of the system). During the training mode, a finite sequence of vectors is repeatedly applied at the SNN input (the training bit sequence) and the task of the network consists in recognizing the sequence: at each time step the SNN has to provide the next bit of the sequence (see the illustration of Fig. 3). The network efficiency is evaluated estimating the probability of the SNN prediction success. At each evaluation

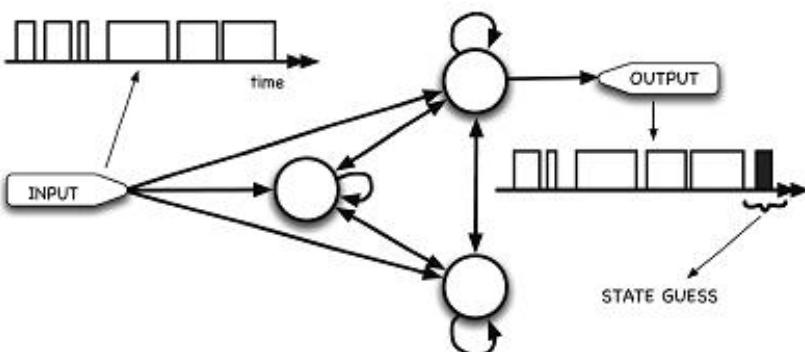


Fig. 3. A complete SNN implements all the possible inter-neuron connections. Such networks are trained to recognize temporal patterns.

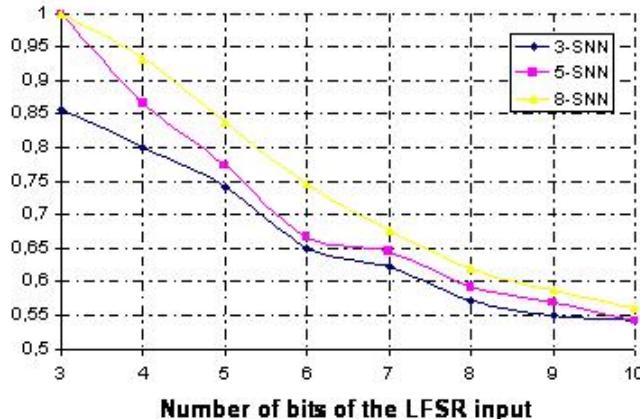


Fig. 4. Success probability of a spiking neural network of 3, 5 and 8 neurons assembled in a complete topology. As can be appreciated, the total “memory” of the system is proportional to the number of neurons of the network.

step a mutated configuration provided by the GAC is used to configure the SNN. The FC bloc evaluates the probability of success of SNN predictions and the mutated configuration is therefore stored or discarded (see Fig. 2).

We configured different networks containing three, five and eight neurons, each one connected to the rest of the network thus assembling a complete topology. In Fig. 3 we show the case with three neurons (defined as a 3-SNN).

The training bit sequence must be as complex as possible to maximize the pattern recognition difficulty. Therefore, we selected the generation of pseudo-random strings provided by LFSR digital blocks. Pseudo-random bit strings are characterized to have the same statistical properties as random sequences with the only characteristic that pseudo-random sequences have a periodicity. In our experiment we used training sequences of $N=7, 15, 31, 63, 127, 255, 511$ and 1023 bits (using LFSR of 3 until 10 bits length). With the selection of this type of pseudo-random sequences, the memorization task difficulty is maximized.

We applied each pseudo-random training sequence to three different SNN with complete topology (using 3, 5 and 8 neurons). Each SNN is configured by the proposed genetic-based self-learning architecture. At each time step, the network has to guess the next bit that will be provided by the LFSR. Once the configuring circuitry has been stabilized to an optimum configuration we evaluate the probability of success associated to this final configuration. In Fig. 4 we provide the different prediction success for each network as a function of the number of bits of the LFSR. It is observed that, as the number of bits of the sequence increases and as the number of neurons decreases the network presents a lower prediction success. We defined a new performance metric of Neural Networks that we refer to as the M-index that measures the ability of the system to recognize a pseudo-random bit sequence. The SNN M-index is defined as *the maximum pseudo-random sequence length that the network is able*

to recognize with a 90% of success probability. This index is a good indicator of the network processing capability, independently on the type and topology of the selected neural system. The M-index of the three networks were 5, 12 and 22 for the 3-SNN, 5-SNN and the 8-SNN respectively.

4 Conclusions

In this work we proposed a simple architecture for SNN self-configuration. The proposed system implements in hardware a simplified genetic algorithm. We applied the proposed architecture to temporal pattern recognition analysis. Different pseudo-random sequences were applied to different networks and the success probability was evaluated. A new performance metric has been developed that may be applied to measure the processing capacity of networks.

Acknowledgments. This work was supported in part by the Balearic Islands Government in part by the Regional European Development Funds (FEDER) and in part by the Spanish Government under projects PROGECIB-32A and TEC2008-04501.

References

1. Malaka, R., Buck, S.: Solving nonlinear optimization problems using networks of spiking neurons. In: Int. Joint Conf. on Neural Networks, Como, pp. 486–491 (2000)
2. Sala, D.M., Cios, K.J.: Solving graph algorithms with networks of spiking neurons. IEEE Trans. on Neural Net. 10, 953–957 (1999)
3. Gerstner, W., Kistler, W.M.: Spiking neuron models. Cambridge University Press, Cambridge (2002)
4. Rosselló, J.L., Canals, V., de Paúl, I., Bota, S., Morro, A.: A Simple CMOS Chaotic Integrated Circuit. IEICE Electronics Express 5, 1042–1048 (2008)

Appendix: VHDL Code of the Digital Spiking Neuron

Each neuron is implemented using a VHDL code in which the connection strength is selected to be a fraction of the neuron threshold. Subsequently we show the VHDL code for a digital neuron with four inputs and the previously described characteristics. Weight is selected to be equal to $\pm 2/5$ (threshold value of 20 for the post-synaptical potential and weight of ± 8).

```
-- VHDL model for a digital Spiking Neuron*****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```

ENTITY neurona IS

PORT -- each input is codified with two bits
(
    clk      : IN      STD_LOGIC;
    in1      : IN      STD_LOGIC;
    in1x     : IN      STD_LOGIC_VECTOR(0 to 1);

    in2      : IN      STD_LOGIC;
    in2x     : IN      STD_LOGIC_VECTOR(0 to 1);

    in3      : IN      STD_LOGIC;
    in3x     : IN      STD_LOGIC_VECTOR(0 to 1);

    in4      : IN      STD_LOGIC;
    in4x     : IN      STD_LOGIC_VECTOR(0 to 1);

    outx     : out      STD_LOGIC
);

END neurona;

```

```

ARCHITECTURE neuron OF neurona IS

BEGIN
canviestat:
PROCESS (clk)
    VARIABLE state: INTEGER RANGE 0 TO 31;
BEGIN
    IF (clk'EVENT) and (clk='1') THEN
        outx<='0';
        IF (in1='1' and in1x="01") THEN
            state:=state+8;
        END IF;
        IF (in1='1' and in1x="10") THEN
            state:=state-8;
        END IF;
        IF (in2='1' and in2x="01") THEN
            state:=state+8;
        END IF;
        IF (in2='1' and in2x="10") THEN
            state:=state-8;
        END IF;
        IF (in3='1' and in3x="01") THEN

```

```
        state:=state+8;
    END IF;
    IF (in3='1' and in3x="10") THEN
        state:=state-8;
    END IF;
    IF (in4='1' and in4x="01") THEN
        state:=state+8;
    END IF;
    IF (in4='1' and in4x="10") THEN
        state:=state-8;
    END IF;
    state:=state-1;
    IF (state>20) THEN
        outx<='1';
        state:=1;
    END IF;
    IF (STATE=0) then
        outx<='0';
    END IF;
END IF;
END PROCESS canviestat;
END neuron;
```

Image Recognition in Analog VLSI with On-Chip Learning

Gonzalo Carvajal, Waldo Valenzuela, and Miguel Figueroa

Department of Electrical Engineering

Universidad de Concepción

{gcarvaja, waldovalenzuela, miguel.figueroa}@udec.cl

Abstract. We present an analog-VLSI neural network for image recognition which features a dimensionality reduction network and a classification stage. We implement local learning rules to train the network on chip or program the coefficients from a computer, while compensating for the negative effects of device mismatch and circuit nonlinearity. Our experimental results show that the circuits perform closely to equivalent software implementations, reaching 87% accuracy for face classification and 89% for handwritten digit classification. The circuit dissipates 20mW and occupies 2.5mm² of die area in a 0.35μm CMOS process.

1 Introduction

Image recognition is the process of assigning classes to instances of a set of acquired images, and constitutes one of the most challenging tasks in image processing and analysis. Many applications of image recognition, such as face and handwriting recognition, are of key importance in portable and embedded systems for the implementation of intuitive user interfaces and secure user authentication. Unfortunately, real-time software implementations of image classification algorithms on embedded processors are incapable of simultaneously meeting the computational requirements of the algorithm and the low power and space constraints of a portable system. Custom analog VLSI circuits can implement moderate-resolution computation using orders of magnitude less power and die area than digital hardware, and are also a good match for the structural regularity of image recognition neural networks. However, the performance of these circuits is limited by signal offsets, device mismatch, charge leakage and nonlinear transistor behavior. Conventional design techniques can reduce these effects, but they increase power and area, rendering analog solutions less attractive.

In this paper, we present alternatives for analog implementation of neural networks for image recognition. Unlike traditional chip-in-the-loop solutions, we use local adaptation and learning to train the network on chip after fabrication with a minimum impact on power and area. We present two hardware dimensionality reduction networks based on Principal Components Analysis (PCA) and one based on Linear Discriminant Analysis (LDA), and two alternatives for classification based on Manhattan-distance metrics in the feature space: nearest

neighbor and Radial Basis Function networks. We show that, combined with circuit calibration techniques, on-chip learning is able to compensate for most of the effects of device mismatch, allowing our circuit to reach 87% accuracy for face classification and 89% for handwritten digit classification, which is comparable to traditional software implementation of the algorithms.

The rest of the paper is organized as follows: Section 2 introduces the algorithms used in our implementation. Sections 3 and 4 describe the hardware used for dimensionality reduction and classification, respectively. Finally, Section 5 shows our experimental results classifying faces and handwritten digits.

2 Subspace-Based Methods for Image Recognition

Using traditional classification algorithms on high-dimensional data results in prohibitively high computational requirements. However, the redundancy present in most images makes it possible to obtain a reduced representation which retains most of the original information in the data. Using this approach, image recognition is composed of two tasks: dimensionality reduction and classification.

Consider an image vector $\mathbf{x}_{n \times 1}$, and the matrix $\mathbf{X}_{n \times N} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ corresponding to a labeled database of N images. Linear subspace methods compute the transformation $\mathbf{Y} = \mathbf{W}^{*T} \mathbf{X}$, where $\mathbf{Y}_{m \times N}$, with $m < n$, is the reduced representation of the database set obtained by projecting the input set \mathbf{X} onto a *feature space* using the transformation $\mathbf{W}_{n \times m}^*$, which is computed to optimize a specific criterion in the feature space.

Two classic linear methods for dimensionality reduction are Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA) [1]. The former constructs \mathbf{W}^* such that its columns are the m largest eigenvectors of the covariance matrix of \mathbf{X} , which maximizes the variance (information) of the projected components of \mathbf{X} in the feature space. LDA takes advantage of labeled data to maximize the distance between classes in the feature space. It defines two matrices \mathbf{S}_W and \mathbf{S}_B representing the scatter (variance) within classes and the scatter between different classes, respectively. In LDA, the columns of \mathbf{W}^* are the m largest eigenvectors of $\mathbf{S}_W^{-1} \mathbf{S}_B$. As result, LDA tends to create clusters for each class in the reduced space to improve classification. Unlike PCA, the number of LDA projections m in the feature space is limited to $m < P$, where P is the number of classes in the labeled training set.

The simplest classification method is nearest-neighbor, which uses a distance metric to select the closest element of the labeled database to the input image in the feature space. Because of its hardware simplicity, in this work we use the Manhattan distance. We also implement and test a more advanced classification method based on Radial Basis Function (RBF) networks, which use a hidden layer of neurons with Gaussian activation functions to detect clusters in the feature space, and are thus particularly well suited to LDA dimensionality reduction.

The next two sections describe the analog hardware implementation of the dimensionality reduction and classification networks.

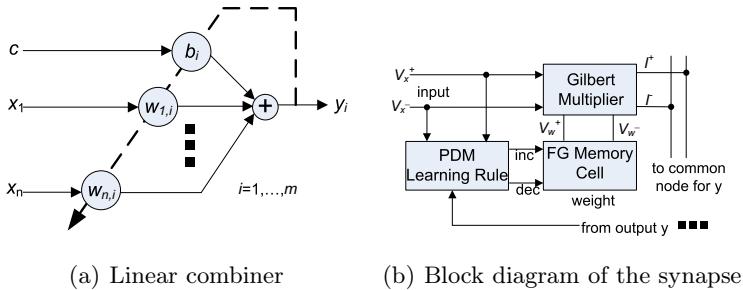


Fig. 1. Adaptive linear combiner in analog VLSI

3 Dimensionality Reduction

The dimensionality reduction network has an identical structure for PCA and LDA, as both compute the product of the input vector by the coefficient matrix, and can be implemented in analog hardware using an array of m n -input linear combiners, with each synapse composed by a multiplier and a memory cell to store a synaptic weight. Fig. 1(a) depicts the architecture of the linear combiner. Because of offsets in the analog multipliers, the output of the linear combiner also exhibits an accumulated offset which depends on the weight values and the properties of the hardware. In order to adaptively compensate for this offset, each combiner features a constant input c connected to a *bias synapse* trained with an anti-Hebbian learning rule [2]. The dashed line going back from the output to the synapses represents a learning rule which adapts the synaptic weights according to the algorithm used to train the network.

Fig. 1(b) shows a block diagram of each individual synapse. A standard analog Gilbert multiplier computes the product of each pixel of the input image, represented as a differential voltage, and the local synaptic weight stored in an analog memory cell. An accurate transformation requires a linear response to the pixel value, and consequently we designed the multipliers to maximize the linearity of that input [3]. The differential output current of each multiplier is summed across the linear combiner by simply connecting their outputs to common wires. An analog memory cell stores the synaptic weight. We implement the memory cell using floating-gate pFET transistors which provide nonvolatile storage, avoiding the charge leakage problems associated with VLSI capacitors, and thus enabling open-loop operation after training. The memory cell, described in detail in [7], uses negative feedback around a small operational amplifier and an on-chip voltage shifter to provide accurate, linear, and bidirectional weight updates with a resolution of up to 12 bits. The updates are performed by applying digital pulses of fixed width to the inputs *inc* and *dec*, and the magnitude of the updates depends linearly on the density (frequency) of the pulses. We implement Pulse-Density Modulation (PDM) learning rules using conventional digital AND and OR gates. Device mismatch in the memory cells results in asymmetric and mismatched learning rates. We compensate for this effect using local calibration

circuits, also based on floating gates, to achieve symmetric and uniform learning rates across the chip with a resolution exceeding 12 bits in a $0.35\mu\text{m}$ CMOS process.

The learning rule implements the method used to compute the coefficient values. A previous paper [3] describes a LMS-based chip-in-the-loop approach, where the coefficients are computed off line and used to generate an external reference to calibrate and write the weights onto the chip. In this paper, we focus on on-chip learning methods, which are suited to cases where the reference image database must be learned during autonomous system operation. The rest of this section describes the implementation of learning rules for two PCA learning rules: Generalized Hebbian Algorithm (GHA) and Adaptive Principal Component Extraction (APEX). The coefficients of LDA must be computed off line and thus cannot be learned directly on chip, however Section 5 includes results for a LDA network where we used the above mentioned on-chip LMS training to compensate for device mismatch when writing the coefficients.

Throughout the analysis presented in the rest of this section, we assume that the inputs have zero mean and have been normalized. Also, for simplicity we assume a linear model for the multipliers given by

$$o = (a_x x + \gamma_x)(a_w w + \gamma_w) \quad (1)$$

where o is the multiplier output, x and w are the inputs, γ_x and γ_w are input offsets, and a_x and a_w are gains associated with each input. These parameters vary across different multipliers due to device mismatch. In Section 5 we present experimental results from real nonlinear multipliers, and include comparisons with chip-in-the-loop methods for PCA and LDA.

Adaptive algorithms for PCA iteratively compute a set of orthogonal weights that maximize the output variance of each individual combiner. In the GHA algorithm [5], output maximization and orthogonalization are performed by a single learning rule:

$$\Delta \mathbf{w}_{j,k} = \mu y_{j,k} \left[\mathbf{x}_k - \sum_{i=1}^j y_{i,k} \mathbf{w}_{i,k} \right] \quad (2)$$

where $y_{j,k} = \mathbf{w}_{j,k}^T \mathbf{x}_k$ is the j -th principal component of the input \mathbf{x} at time k . The bracketed term represents the reconstruction error of \mathbf{x} from the first j components. The GHA rule for the j -th component seeks to minimize the local reconstruction error for the fraction of \mathbf{x} that has not been yet approximated by the previous components.

The APEX algorithm [5] combines an output maximization network with inhibitory synapses for weight orthogonalization. Now, the output of the j -th combiner is $y_{j,k} = \mathbf{w}_{j,k}^T \mathbf{x}_k - \sum_{i=1}^{j-1} l_{ij,k} y_i$ and the weights adapt with the rules:

$$\Delta \mathbf{w}_{j,k} = \mu y_{j,k} (\mathbf{x}_k - y_{j,k} \mathbf{w}_{j,k}) \quad (3)$$

$$\Delta l_{ij,k} = \mu y_{j,k} y_i \quad \text{with } i < j \quad (4)$$

where l_{ij} is the inhibitory weight connecting the output of the i -th combiner to the j -th combiner. The learning rule for the first component is identical for both algorithms:

$$\Delta \mathbf{w}_k = \mu y_k (\mathbf{x}_k - y_k \mathbf{w}_k) = \mu y_k (\mathbf{x}_k - \mathbf{x}'_k) \quad (5)$$

where μ is the learning rate and \mathbf{x}'_k is the instantaneous reconstruction of \mathbf{x} . We now derive an implementation of the learning rule that minimizes the effects of device mismatch.

Let us consider an analog linear combiner such as the one depicted in Fig. 1(a) computing the first principal component y of \mathbf{x} . Using the multiplier model of Eqn. (II), the distorted output will be computed as:

$$\bar{y}_k = \mathbf{x}_k^T (\mathbf{A}_x \mathbf{A}_w \mathbf{w}_k + \mathbf{A}_x \boldsymbol{\gamma}_w) + [\boldsymbol{\gamma}_x^T (\mathbf{A}_w \mathbf{w}_k + \boldsymbol{\gamma}_w) + c(ab b_k + \gamma_b)] \quad (6)$$

where $\mathbf{A}_x = \text{diag}([a_{x_1} \dots a_{x_n}])$, $\mathbf{A}_w = \text{diag}([a_{w_1} \dots a_{w_n}])$, $\boldsymbol{\gamma}_x = [\gamma_{x_1} \dots \gamma_{x_n}]^T$, $\boldsymbol{\gamma}_w = [\gamma_{w_1} \dots \gamma_{w_n}]^T$, a_b , and γ_b represent the gain and offset of each multiplier, which are unknown at design time. Upon convergence, the bracketed term in Eqn. (6) will be a constant offset, which can be cancelled by a bias synapse trained with the anti-Hebbian rule $b_{k+1} = b_k - \mu c y_k$. Thus, Eqn. (6) reduces to:

$$\bar{y}_k = \mathbf{x}_k^T (\mathbf{A}_x \mathbf{A}_w \mathbf{w}_k + \mathbf{A}_x \boldsymbol{\gamma}_w) = \mathbf{x}_k^T \bar{\mathbf{w}}_k \quad (7)$$

From Eqn. (II) and (5), \mathbf{x}'_k is computed as:

$$\bar{\mathbf{x}}'_k = \bar{y}_k (\mathbf{A}_y \mathbf{A}'_w \mathbf{w}_k + \mathbf{A}_y \boldsymbol{\gamma}'_w) + \boldsymbol{\gamma}_y (\mathbf{A}'_w \mathbf{w}_k + \boldsymbol{\gamma}'_w) \quad (8)$$

where \mathbf{A}_y , \mathbf{A}'_w , $\boldsymbol{\gamma}_y$, and $\boldsymbol{\gamma}'_w$ are the gains and offsets of the associated multipliers. Replacing Eqns. (7) and (8) into Eqn. (2), we determine the effective learning rule modified by device mismatch:

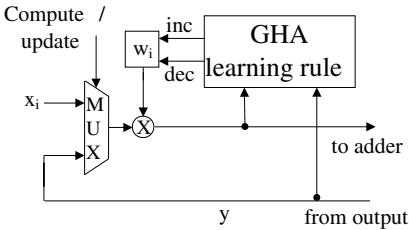
$$\overline{\Delta \mathbf{w}}_k = \mu \bar{y}_k (\mathbf{x} - \bar{y}_k (\mathbf{A}_y \mathbf{A}'_w \mathbf{w}_k + \mathbf{A}_y \boldsymbol{\gamma}'_w)) = \mu \bar{y}_k (\mathbf{x} - \bar{y}_k \bar{\mathbf{w}}'_k) \quad (9)$$

where the last (constant) term of Eqn. (8) is omitted because \bar{y}_k has zero mean.

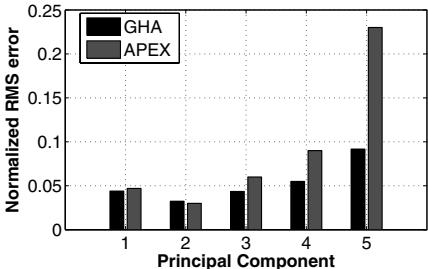
The key to compensating for the effects of mismatch expressed above is to use the same analog multipliers to compute \bar{y}_k and $\bar{\mathbf{x}}'_k$. In this case $\mathbf{A}_x = \mathbf{A}_y$, $\mathbf{A}_w = \mathbf{A}'_w$, and $\boldsymbol{\gamma}_w = \boldsymbol{\gamma}'_w$, and the learning rule becomes:

$$\overline{\Delta \mathbf{w}}_k = \mu \bar{y}_k (\mathbf{x} - \bar{y}_k \bar{\mathbf{w}}_k) \quad (10)$$

where \bar{y}_k and $\bar{\mathbf{w}}_k$ are the modified weight and output defined in Eqn. (7). Eqn. (10) is equivalent to the original rule in Eqn. (5), but with a new weight vector modified by device mismatch. A convergence analysis similar to LMS [2] demonstrates that this weight converges to a value that compensates for the effects of mismatch in the computation of the output of Eqn. (6). Note that this will only be the case if the same multipliers are used to compute y_k and \mathbf{x}'_k . Fig. 2(a) shows a block diagram for one individual synapse used in both algorithms. The multiplexer shares the analog multipliers between the computation of y_k and \mathbf{x}'_k , and is controlled by a digital signal that alternates its value during the computation and adaptation phases of the algorithm.



(a) GHA/APEX synapse



(b) Output error of PCA network

Fig. 2. Synapse for on-chip PCA learning and measured output error

We use a natural extension of the design in Fig. 2(a) for the higher-order outputs of GHA and the Hebbian synapses of APEX. The inhibitory lateral connections of APEX use an anti-Hebbian learning rule which naturally compensates for offset and gain mismatch [2].

Fig. 2(b) depicts experimental results from our implementations of GHA and APEX in a $0.35\mu\text{m}$ CMOS process. The figure plots the normalized Root-Mean-Square error of both networks for the first 5 principal components using the ORL face database. Even though both algorithms successfully compensate for most of the effects of device mismatch (without compensation, the algorithms simply fail to converge), they still exhibit a measurable distortion in the computation of the principal components. Due to its higher structural complexity, APEX is more sensitive to these effects, especially in the higher-order components. However, the experiments presented in Section 5 show that this distortion does not significantly affect the classification performance of the network on real images.

4 Classification

As discussed in Section 2, we evaluate two classification methods: nearest neighbor and Radial Basis Function (RBF) networks. Because of its implementation simplicity, both methods use the Manhattan distance, defined for two vectors $\mathbf{x} = [x_1 \dots x_n]$ and $\mathbf{y} = [y_1 \dots y_n]$ as $d = \sum_{i=1}^n |x_i - y_i|$. Fig. 3(a) illustrates the distance computation circuit for the nearest-neighbor method. Each projection of the training set onto the feature space is stored as a current in an analog memory cell, simpler and smaller than the cell used in the dimensionality reduction network, and written using a self-limiting write process to compensate for device mismatch. The difference between each projection of the pattern and the test input is computed by inverting the polarity of one of the signals and adding the currents. A current comparator based on a simple transconductance amplifier determines the sign of the result and uses a 2×2 crossbar switch to invert the polarity of the outputs and thus computes the absolute value. A current-based Winner-Take-All (WTA) circuit [6] selects the database element with the smallest distance to the test input, and assigns its class to the test image.

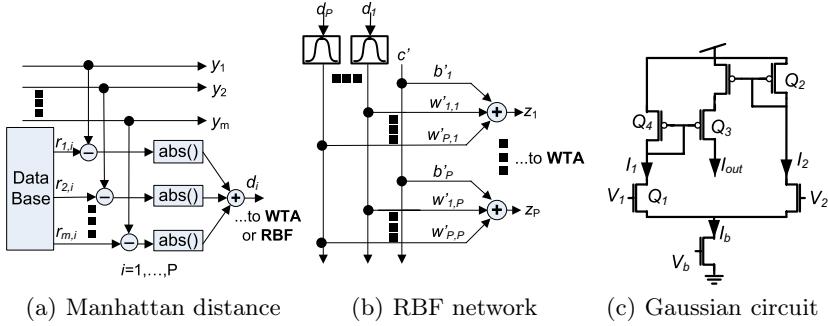


Fig. 3. Architecture and circuits for classification. d_i is the computed distance in the feature space between the feature vector \mathbf{y} and a stored database element \mathbf{r}_i , and z_j are the outputs of the RBF network.

A classification structure based on RBF networks, proposed in [8], provides a representation for the clusters assigned to each image class. This method has a compact implementation because it only stores and compares the parameters of each class in the training set, while nearest-neighbor considers each separate element. Moreover, because the LDA algorithm specifically creates clusters in the feature space, it is well suited to this classification approach. Because we know in advance the number of clusters and can compute their centers and widths, the design of the network is considerably simplified. Fig. 3(b) shows the architecture of the two-layer RBF network. A circuit similar to Fig. 3(a) computes the Manhattan distance between the test image in the feature space and the centers of each cluster, also stored in small analog memory cells. An array of *bump* circuits [4], shown in Fig. 3(c), compute Gaussian-like functions on each distance. The outputs of the Gaussian functions feed the output layer, composed of linear combiners. Using an on-chip implementation of the LMS algorithm [3], we train each combiner to identify one specific image class in the database. Our LMS learning rule is both compact and highly robust to the effects of device mismatch, allowing for open-loop operation with about 10 bits of resolution. A current-based WTA circuit selects the largest output z_t and assigns a class label t to the test image.

5 Experimental Results

We designed and fabricated analog multipliers, memory cells, and weight-update rules to implement the building blocks described in the previous section using a $0.35\mu\text{m}$ CMOS process. We tested the fabricated chips in the lab and developed a software emulator that allows us to test the static performance of an arbitrary network with less than 0.5% error compared to the fabricated hardware. We used this emulator to test the performance of different network configurations on a face recognition task using the Olivetti Research Labs (ORL) database, composed

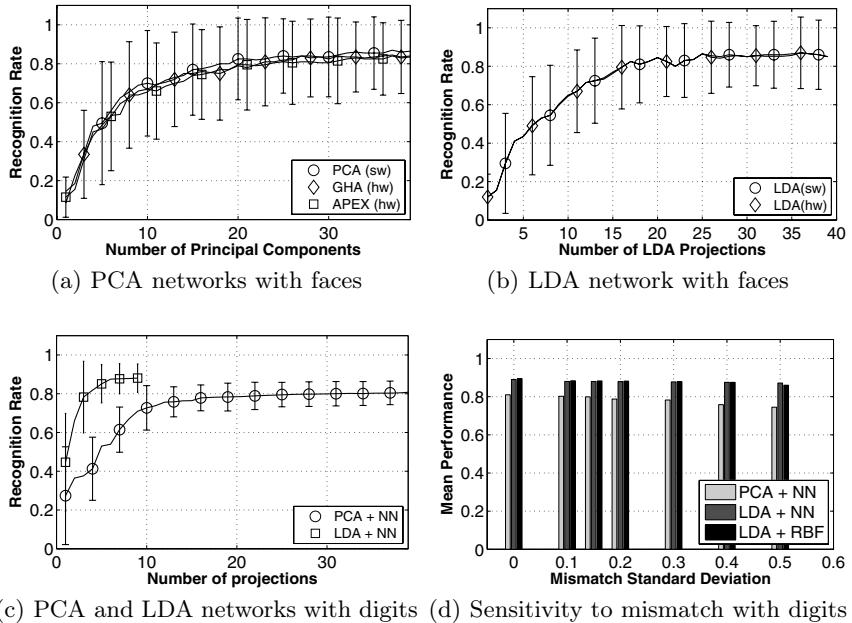


Fig. 4. Performance in classification of faces and handwritten digits

of 10 photos of 40 different subjects. We randomly picked 5 photos of each subject to build the training set, and used the other 5 to test the performance of the networks. Due to the high computational cost of the emulation, we had to reduce the size of the images to 14x14 pixels. We trained the networks with 5000 presentations of the 200 training set.

Fig. 4(a) plots the mean classification performance for 40 subjects with the GHA and APEX circuits using the nearest-neighbor algorithm, as a function of the number of principal components used in the classification. As a reference, we also plot the performance of a software implementation of PCA using eigenvalue analysis to compute the coefficients. The error bars show one standard deviation above and below the mean. The classification performance of the software implementation peaks at 86%, while the hardware networks with trained on-chip with GHA and APEX peak at 81% and 79%, respectively. These numbers show that the distortions introduced to the principal components by the analog network mostly preserves the separability of the classes. The large error bars are due to the small number of samples in the test set. The estimated power consumption of the circuit with 196 (14x14) inputs and 39 projections is 20mW (600nJ per classification with 30 μ s settling time), and the layout area is 2.5mm². These numbers represent a 5x reduction in area and more than 100x reduction in power compared to standard cell-based digital implementations [10, 9].

We then tested the performance of a hardware LDA network with nearest-neighbor classification on the ORL face database. Because it is not possible to

compute the LDA coefficients adaptively, we used a chip-in-the-loop approach where we computed the coefficients in software and downloaded their values onto the chip. To compensate for offsets and mismatch in the multipliers and memory cells, instead of writing the coefficients directly we trained the dimensionality reduction network using supervised learning with an on-chip implementation of the LMS algorithm [3]. The graph in Fig. 4(b) shows the classification performance of the calibrated LDA network as a function of the number of components used in the feature space. We also include a software version of LDA as a reference. The hardware LDA network follows closely the performance of the software, with both peaking at 87% for 39 components with a smaller standard deviation than PCA. Because the implementation of LMS is more compact than GHA and APEX, this circuit occupies a die area of only 1.8mm^2 . The power dissipation and classification times are the same as in the PCA networks. Using 45×45 -pixel images, the PCA and LDA hardware networks (with coefficients computed off-chip and trained with the LMS algorithm) reach a classification performance of 90% and 95%, respectively, which matches closely that of equivalent software implementations.

We also tested the performance of our network on a United States Postal Service database of 11,000 16×16 -pixel images of digits from handwritten ZIP codes. We used one half of the database for training and one half for testing. Fig. 4(c) shows the performance of hardware PCA and LDA networks as a function of the number of projections used in the feature space. In both cases, we computed the coefficients off-chip and trained the dimensionality reduction network on-chip with the LMS algorithm. The performance peaks at 80.5% for PCA with 39 projections and at 89% for LDA with 9 projections (LDA is limited to 9 projections for 10 classes).

Finally, we evaluated the sensitivity of our networks to mismatch in the classification stage. Using Spice simulations, we varied transistor parameters in the classification circuits using a Gaussian process, and measured the performance of the network. Fig. 4(d) shows the classification performance of handwritten digits for the PCA and LDA networks with nearest-neighbor selection, and the LDA network with an RBF classifier (PCA does not naturally form clusters in the feature space and is thus not adequate for RBF). The figure shows that PCA is clearly more sensitive to mismatch, reducing its performance from 80.5% to 74.4% as the standard deviation of the mismatch parameters varies from 0 to 0.5. On the other hand, LDA clusters the data so that it is not significantly affected by mismatch: classification performance with nearest neighbor degrades from 89% to 87.1%, while RBF classification varies from 89.5% to 86.2%.

6 Conclusions

We presented the design of an analog-VLSI system for image recognition using a dimensionality reduction network and a classification stage. We use PCA and LDA to train the dimensionality reduction network off-chip and program the coefficients onto the chip using supervised learning with LMS to compensate for

the effects of device mismatch and circuit nonlinearities. We also implemented local adaptation rules to learn the PCA coefficients on chip using the GHA and APEX algorithms. The classification stage uses Manhattan distances and nearest-neighbor selection or RBF networks to assign classes to test images. Our experimental results on face and handwritten digit recognition show that chip-in-the-loop learning with LMS achieves the same classification performance as a software implementation of the algorithms. On-chip learning with GHA and APEX are outperformed by the software by 5%-7%, but have the advantage of being trained autonomously after being deployed in the field. Because it maximizes separation within classes, LDA is more robust to device mismatch in the classification stage than PCA, but it requires off-chip training. We are currently working on the design of a reconfigurable block to implement multiple learning rules, support for inputs of higher dimensionality, and integration with an embedded microprocessor.

References

1. Belhumeur, P., Hespanha, J., Kriegman, D.J.: Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(7), 711–720 (1997)
2. Carvajal, G., Figueroa, M., Bridges, S.: Effects of Analog-VLSI Hardware on the Performance of the LMS Algorithm. In: International Conference on Artificial Neural Networks (ICANN), Athens, Greece, September 10-14, pp. 963–973 (2006)
3. Carvajal, G., Valenzuela, W., Figueroa, M.: Subspace-Based Face Recognition in Analog VLSI. In: Advances in Neural Information Processing Systems 20, pp. 225–232. MIT Press, Cambridge (2008)
4. Delbruck, T.: Bump circuits for computing similarity and dissimilarity of analog voltages. In: Proc. of International Joint Conference on Neural Networks, Washington, DC, USA, pp. 475–479. IEEE Computer Society, Los Alamitos (1991)
5. Diamantaras, K.I., Kung, S.Y.: Principal Component Neural Networks: Theory and Applications. Wiley Interscience, Hoboken (2001)
6. Donckers, N., Dualibe, C., Verleysen, M.: Design of complementary low-power CMOS architectures for looser-take-all and winner-take-all. In: MICRONEURO 1999: Proc. of the 7th Intl. Conf. on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems, Washington, DC, USA, p. 360. IEEE Computer Society, Los Alamitos (1999)
7. Figueroa, M., Bridges, S., Diorio, C.: On-chip compensation of device-mismatch effects in analog VLSI neural networks. In: Advances in Neural Information Processing Systems 17. MIT Press, Cambridge (2005)
8. Oh, B.-J.: Face recognition using radial basis function network based on lda. In: IEC (Prague), pp. 255–259 (2005)
9. Prasanna, C.S.S., Sudha, N., Kamakoti, V.: A Principal Component Neural Network-Based Face Recognition System and Its ASIC Implementation. In: VLSI Design, pp. 795–798 (2005)
10. Shams, N., Hosseini, I., Sadri, M., Azarnasab, E.: Low Cost FPGA-Based Highly Accurate Face Recognition System Using Combined Wavelets With Subspace Methods. In: IEEE International Conference on Image Processing, pp. 2077–2080 (2006)

Behavior Modeling by Neural Networks

Lambert Spaanenburg¹, Mona Akbarniai Tehrani^{1,*},
Richard Kleihorst², and Peter B.L. Meijer²

¹ Lund University, Dept. of Electrical and Information technology,
P.O. Box 118, 22100 Lund, Sweden

lambert@eit.lth.se, sx06mai@student.lth.se

² NXP Semiconductors, Research,
High Tech Campus, 5656 AE Eindhoven, The Netherlands
richard.kleihorst@vito.be, Peter.b.l.meijer@nxp.com

Abstract. Modeling of human and animal behavior is of interest for a number of diagnostic purposes. Convolutional neural networks offer a constructive approach allowing learning on a limited number of examples. Chaotic tendencies make that learning is not always successful. The paper looks into a number of applications to find the reason for this anomaly and identifies the need for behavioral references to provide determinism in the diagnostic model.

Keywords: Behavioral modeling, convolutional neural network, collision avoidance, 3-tier architecture, Sinai billiard system.

1 Introduction

Neural networks have received much interest for image understanding and intelligent control. Some work has been done in categorizing social behavior of humans and animals, largely from a biological point of view to understand and model how nature works. Of late, homeland security issues kindle renewed interest. Many public places have been equipped with cameras, but the mere amount makes on-line monitoring by humans impossible. This has created the interest to automatically detect and diagnose suspicious behavior.

Smart vision sensors for automated surveillance of public places can be based on Commercially-Off-The-Shelf (COTS) processors, but specialized image processors are required for massively crunching pixels into features while achieving real-time performance at a low power budget. This poses additional restrictions on the implementation of the neural networks. Fully connected, monolithic neural networks have a limited capacity when implemented as a single digital ASIC, while they tend to give cache problems on COTS processors and usually do not agree with the fixed-point number representation popular for the cheaper

* This paper is published with the support of Image Processing and Interpolation (IPI) department of Gent University, Gent, Belgium.

embedded platforms. This pleads for the use of heterogeneous platforms such as WiCA [1], where a conventional 8051 is combined with the IC3D ASIC, a 320-core pixel cruncher. As monolithic neural networks are ill-famed for their learning problems, hierarchical structures seem advisable, leaving the question: how to build and distribute this hierarchy on the platform.

This sets the case for the convolutional neural network, a layered hierarchy of mostly-neural modules. The paper is therefore composed as follows. First we introduce the modular hierarchy of neural networks with an illustrative application. In section 3 we discuss automotive collision avoidance and in section 4 we treat the Sinai billiard system. Finally we discuss our findings and draw some conclusions.

2 The Modular Hierarchy

The core problem in developing neural networks is collecting, validating and presenting the examples. Often it is quoted that more than 90% of the project time has been devoted to getting and preparing the data set to learn from [2]. This is the more surprising, as usually a large amount of ground knowledge exists such as basic physical laws that can be incorporated in several ways.

Firstly the networks may be initialized to approximate “ideal” behaviour. The clean network has weight values that are randomly selected small values around zero. This is meant to create no built-in preferences, so that anything can be learnt. Larger and non-random weight values will introduce preferences, as desired, but it is not always transparent how they can be calculated.

Secondly, the problem can be described by more abstract examples. To facilitate this, the abstract notion must be founded on reality by known principles, the so-called ground truth. Although the relation is well-defined, the use of fixed knowledge can aggravate the learning problem [3]. Often it is cast into a neural network so its parameters can be slightly adapted to the circumstances under which the measurements are made.

For large networks, a monolithic solution tends to be hard (if not impossible) to train [4]. This is caused by conflicting data in the training set or by data that becomes conflicting due to the training itinerary. A solution is to compose the network from modules in the form of sub-networks that can be individually initialized or trained. A fortunate side effect is the structuring and reduction of the example set. Sometimes the composition can be derived from the problem at hand [5], but the convolutional network provides a generic framework as a hierarchy of layers with parallel operating sub-networks.

The layered hierarchy is based on the combined operation of many small fully-connected feed-forward neural networks. The underlying idea is a 3-tier architecture, as popularized in many fields of telecommunication (Figure 1) with the following meaning:

- The lower (*foundation*) tier expresses the basic technological ingredients. This may introduce facts of common knowledge to the neural system, abstract fundamental physical parameters to the digital realm, or unify different graphic

packages into a single programming interface. In image understanding, we find here the low-level pixel operations for pre-processing, built from initialized sub-networks.

- The middle (*processing*) tier contains the operational functions. It provides the transformations and operations to support a domain of applications in some related technological fields. This may offer a set of classes for the modelling in the envisaged domain and/or a set of algorithms to perform numerical support. In image understanding, we find here sub-networks individually trained for a choice of medium-level blob operations.
- The final (*application*) tier provides the interface to the application at hand. It personalizes the domain to provide a direct support to the user. The functionality is expressed in terms of the processing functions. As a consequence, any changes in the application will not induce a major effort as long as they can still be expressed in the available functions. In image understanding, we find here the high-level feature operations trained for the overall problem.

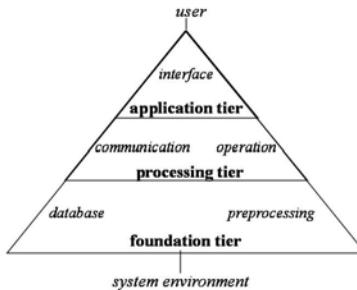


Fig. 1. The 3-tier concept

The introduction of 3 layers aims to decouple platform restrictions and hardware dependencies from the user application. In the software arena, the 3-tier architecture is directly related to concepts of re-use and maintenance; for hardware, the tiers image the integration stages between components and product. For the benefits harvested in applying the multi-tier architecture in neural systems, see [6].

This method has been first applied at full length for the detection of surface defects on steel from a rolling mill [7]. Here, the defects appear as peaks in a flat area. By their reliance on domain rather than application properties, the lower tier can be built from one single network type. Basic physical laws on light reflection are turned into a set of artificial data. Samples of light reflections in all possible gradients are used, varying around the target value characteristic for the specific sub-network. The outputs are the characteristic values corresponding to the applied physical laws but in a manner that makes the sub-networks act as weak classifiers. As the upper networks combine these characteristic values to make the final decisions, it is required that the data to train the lower tier cover the entire problem space (Figure 2).

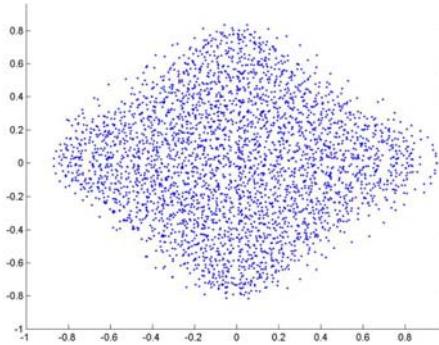


Fig. 2. Plot of target values for the lower networks. The training data represent the whole problem space.

The middle tier serves to provide parameter invariance. Rotation and size-invariance is achieved by arranging the outputs from the lower tier into so-called *peeling-layers* [2]. Width invariance occurs as a large peak and will fill all the three peeling-layers but a small peak will fill only the inner layer. We use the directions of the vectors that result from the lower layer and calculate three properties that reflect continuity in assembling these vectors into potential peaks. Additionally a threshold is applied on the results to suppress any peak detection of a structure that does not adequately reflect a peak. The three output values from the middle networks are used as input for the upper network. The 3rd tier network makes the final decision of whether the larger image is centred over a peak.

In this application, the neural system applies only to single images. It supports the detection of and measurements on defects by intelligently matching the expectations of light reflections to the actual observations. We call this ‘structural stitching’ in contrast to the ‘situational stitching’ to enforce continuity on the assembly that we discuss in the next sections.

3 Collision Avoidance

In the early 90’s the European automotive industry started their first collaborative research program, called Prometheus. The next example comes from the PRO-CHIP project within that program. In the original set-up a large collection of data on a driving car was made. This collection is by itself not usable as it describes mostly just driving along the road without changes in speed and/or direction. Data editing seems necessary to eliminate catastrophic forgetting in adaptive networks, or prevent implicitly giving disproportionately high weight to simple behaviours in non-adaptive networks. For example, driving straight-ahead for a long time tends to destroy all the knowledge about taking a curve. This editing on the set can be a long and cumbersome process, as a lot of tests are necessary to check on the sensitivity of the results to the partial removal of training data.

A better approach is to define a closed set of typical driving situations, from which during learning a random selection with small random variations is made (Figure 3). Different driver characteristics can be obtained by judiciously balancing the number of specific examples in the training set. Such behaviour has been tested in a simulated environment with a variety of uncontrollable disturbances, such as side-wind while turning on a slippery road [8].

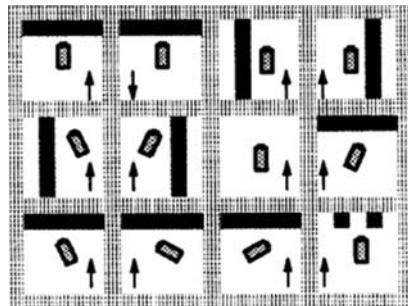


Fig. 3. Basic driving situations (not shown is the 13th situation, where no blockade is near)

A neural network is a non-linear system. Such a system may show chaotic behaviour because of sensitivity for initial conditions. In the driver model this will lead to different behaviour for every different starting position of the car. It is interesting to observe that this is true on the detailed level but on the more abstract level of reasoning as introduced by the driving examples in Figure 3, it has only occurred when the driver had not to take a curve or carefully avoid obstacles, such as the side of the road. Why is that? The answer is that the curves and blockades put the driver a straitjacket on. There are far fewer ways to leave the curve unharmed than to enter it. In other words, the curves serve as points of reference for path convergence such that with every curve accuracy gets largely restored. This becomes visible in car behaviour when comparing a route with an occasional curve with a route with many curves. For instance, after a U-turn all cars are almost at the same trajectory. The same reasoning applies to obstacle avoidance. This has two important consequences. First of all, it seems that trajectories are a meaningful signature of behaviour. The same reasoning applies to obstacle avoidance. This has two important consequences. First of all, it seems that trajectories are a meaningful signature of behaviour. Secondly, it seems that behaviour funnels acting as points of reference can restrict chaotic systems. In the experiments, both the simulation as the limited precision implementation has shown the same repeatable and deterministic trajectories. Figure 4 shows a typical testing ground; further demonstration of continuous correct driving was delivered to the 1991 PRO-CHIP meeting in Darmstadt.

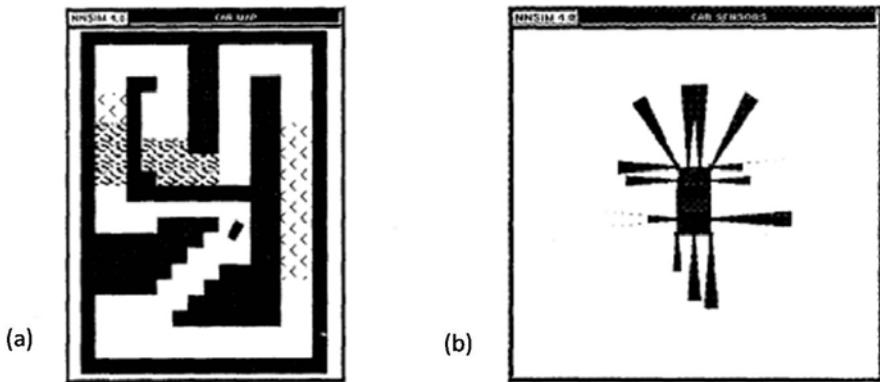


Fig. 4. (a) Screen showing a driving environment with wind (arrows) and ice (shaded roads) and (b) screen showing the actual position, direction und range of the thirteen distance sensors on the car

4 A Billiard System

Based on the aforementioned experience we now simulate a so-called “Sinai billiard table” on an embedded computer vision system with limited dynamic range in its number formats. The Sinai billiard system was introduced by Yakov G. Sinai and shows long-term unpredictability in a non-linear dynamic system with unbounded accuracy [9].

In this system a (simulated) billiard ball is rolling on a square table in straight lines and changes its direction when it hits the walls, corners or the circle in the middle of the table according to the usual laws (angle of incidence equals angle of reflection).

Having the position of the ball, its speed and direction and the knowledge of its environment (the distance to the closest obstacle), the motion pattern can be learnt in different situations separately as the ball faces different obstacles. The input flow vector to the selected neural network (NN) consists of the ball

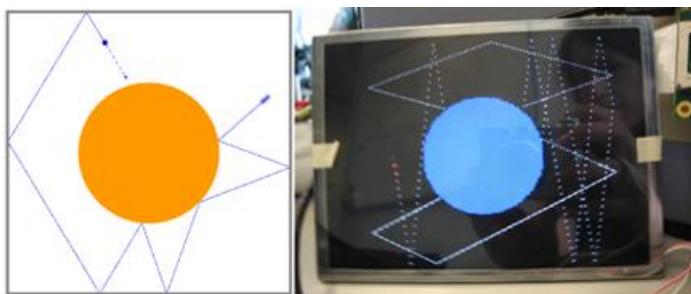


Fig. 5. Sinai billiard ball simulator

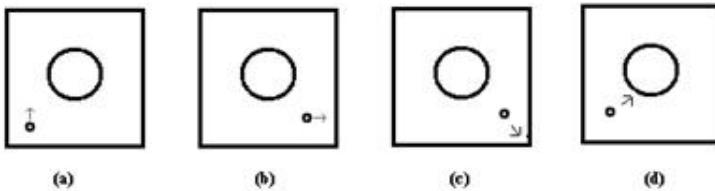


Fig. 6. Different motion patterns of the ball

coordinates in two subsequent frames $f = \{x(t-2), x(t-1), y(t-2), y(t-1)\}$. In addition, coordinates of the ball in the next frame $T = \{x(t), y(t)\}$ are given to the networks as target outputs.

In the first case, a multilayer feed-forward neural network (FFNN) learns the linear movement of the ball (Figure 6a), which can be expressed by the formulas:

$$x[t] = 2 \times x[t-1] - x[t-2] \text{ and } y[t] = 2 \times y[t-1] - y[t-2].$$

A FFNN with a 4-2-2 topology is selected and trained offline using supervised learning with the training data set prepared using a MatLab model of the Sinai billiard ball simulator. It is typical for a convolutional network that the local problem admits a simple construction (here two linear neurons would suffice), but a more generic set-up achieves an easier integration within the overall system. After training in NEUREKA [10], the network has a good generalization results and acceptable error rate (Table 1).

Table 1. Training and testing results for linear movement pattern

Train dataset size	Training error	Test dataset size	Nr. epoch	Test error	Activation function
200	0.08%	200	3000 iterations	0.09%	Sigmoid Function

For the next condition, a FFNN with similar topology is trained to learn the ball motion pattern after collision with the wall and corners (Figure 6b and c).

$$m[t] = 2 \times m[t-1] - m[t-2]. \quad (1)$$

$$n[t] = n[t-2] \quad (2)$$

Here $m = y$ and $n = x$ when the ball collides with vertical walls and $m = x$ and $n = y$ when ball collide with horizontal walls. Error rate in both training phase and test phase is small and the networks learn the motion pattern successfully (Table 2).

Finally, the motion pattern of the ball after collision with the circle in the middle of the table is modelled by another FFNN, with 4-6-4-2 topology. Here the ball follows the following nonlinear pattern:

$$Dist = \sqrt{(Xc - x[t-1])^2 + (Yc - y[t-1])^2} \quad (3)$$

$$nx = (x[t-1] - Xc)/Dist; \quad ny = (Yc - y[t-1])/Dist \quad (4)$$

Table 2. Training and testing results for border collisions

Train dataset size	Training error	Test dataset size	Nr. epoch	Test error	Activation function
200	0.1%	200	4000 iterations	0.09%	Sigmoid Function

$$dp = -nx \times dx[t-1] + ny \times dy[t-1] \quad (5)$$

$$dx[t] = 2 \times dp \times nx + dx[t-1]; \quad dy[t] = -2 \times dp \times ny + dy[t-1] \quad (6)$$

$$x[t] = x[t-1] + dx[t]; \quad y[t] = y[t-1] + dy[t] \quad (7)$$

All the above networks are trained and tested successfully with an error rate of around 1% (Table 3). This larger error is caused by training this network as a single structure instead of as an assembly of 2 networks, which it actually is. Therefore further improvement is easily attainable.

Table 3. Training and testing results for circle wall collisions

Train dataset size	Training error	Test dataset size	Nr. epoch	Test error	Activation function
50	0.16%	50	11000 iterations	1.4%	Sigmoid Function

5 Discussion

In the next step the networks created above are transferred and implemented on the WiCA platform. Putting everything on a single resource, i.e. the 8051 on the WiCA platform has two drawbacks [1]. Firstly, the execution of the neural networks is now based on fixed-point values and therefore precision will become an issue. Secondly, the 8051 is slow and the results only become available once per 2 frames while running the system at 30 fps with VGA resolution. However, the accelerated pixel crunching through the IC3D helps to reduce complexity by representing a blob (like a billiard ball) by only its Centre-of-Gravity coordinates.

During the testing phase the input values of the networks are calculated by parallel pixel processing on the IC3D and coordinate transfer to the 8051. As the position of the ball is known relative to the obstacles defined in the simulator, one of the above NNs is selected in each frame for predicting the next position of the ball, and input values are forwarded to that NN. The results are then sent back to the IC3D for further analysis and live monitoring.

In Figure 7 the results of applying different NNs are shown. White dots on the screen illustrate the mismatch between actual ball position and predicted position by NN (abnormal motion behaviour). Figure 7a represents the situation where only one NN is used to predict linear motion of the ball. In Figure 7b also the second NN is applied which predicts the position of the ball after collision with the border walls. In Figure 7c the third NN is added, predicting position of the ball after collision with the circle in the middle of the screen. In some places faulty mismatches occur due to modelling errors and also because of random



Fig. 7. Results of applying different FFNN to simulator

errors, which are enforced occasionally in the simulator to overcome numerical problems caused by the use of fixed-point values with limited dynamic range.

Observation of the performance and results of the above system lets us identify the problems and challenges facing NN implementations in an embedded platform [1].

- Offline learning and model generation outside the embedded platform as implied by the convolutional network architecture allows optimized modelling for the platform.
- Depending on the word length of the input values and the size of the selected neural network, the outputs may become available with different delays due to different processing times, making it more difficult to have uniform results for monitoring and further analysis.
- The collisions with the wall do not provide a convergence reference for the subsequent behaviour and therefore chaotic movement (drift) sets in. This becomes even more apparent for situations where the round ball collides with a round object that amplifies small differences in angles of incidence.



Fig. 8. A set of observed walking trajectories

All by all, we have a movement that is easy to predict locally, but hard over longer periods, and that will remain so even when more computational precision is available. Even though a normal surveillance system monitoring human movement will have less chaotic moving elements, a convergence reference will be needed. We find this also in modelling larger-scale walking patterns. Figure 8 demonstrates that over some distance the walking patterns can be adequately modelled by carefully stitching the micro-level motion details together while raising the accuracy of the collisions. After some time, remaining chaotic effects of the collisions in the movement have died out and the pattern can be used again for prediction over larger distances.

References

1. Kleihorst, R., Schueler, B., Danilin, A., Heijligers, M.: Smart camera mote with high performance vision system. In: Digest DSC 2006 (2006)
2. Schuermann, B.: Applications and Perspectives of Artificial Neural Networks, VDI Berichte, vol. 1526, pp. 1–14 (2000)
3. Jansen, W.J., Diepenhorst, M., Nijhuis, J.A.G., Spaanenburg, L.: Assembling engineering knowledge in a modular multilayer Perceptron neural network. In: Digest ICNN 1997, pp. 232–237 (1997)
4. Auda, G., Kamel, M.: Modular Neural Networks: A Survey. *J. of Neural Systems* 9, 129–151 (1999)
5. terBrugge, M.H., Nijhuis, J.A.G., Spaanenburg, L., Stevens, J.H.: CNN Applications in toll driving. *J. of VLSI Signal Processing* 23(2/3), 465–477 (1999)
6. Nguyen, C.T.: Method and system for converting code to executable code using neural networks implemented in a very large scale integration (VLSI) integrated circuit, U.S. Patent 6,578,020 (2003)
7. Grunditz, C., Walder, M., Spaanenburg, L.: Constructing a neural system for surface inspection. In: Proc. IJCNN, vol. III, pp. 1881–1886 (2004)
8. Nijhuis, J., Hofflinger, B., Neusser, S., Siggelkow, A., Spaanenburg, L.: A VLSI implementation of a neural car collision avoidance controller. In: Proc. IJCNN, vol. 1, pp. 493–499 (1991)
9. Sinai, Y.G.: Dynamical Systems with Elastic Reflections. *Russian Math. Surveys* 25, 137–191 (1970)
10. Meijer, P.B.L.: Neural Network Applications in Device and Subcircuit Modeling for Circuit Simulation, PhD thesis, Eindhoven University (1996)
11. Tehrani, M.A.: Abnormal motion detection and behaviour prediction, M.Sc. thesis, Lund University (2008)

Statistical Parameter Identification of Analog Integrated Circuit Reverse Models

Bruno Apolloni¹, Simone Bassis¹, Cristian Mesiano¹, Salvatore Rinaudo²,
Angelo Ciccazzo², and Angelo Marotta²

¹ Dept. Computer Science & Advanced Applied Mathematical and Statistical Sciences
via Comelico 39/41, 20135 Milano, Italy

{apolloni,bassis,mesiano}@dsi.unimi.it

² STMicroelectronics

Stradale Primo Sole 50, 95121 Catania, Italy

{salvatore.rinaudo,angelo.ciccazzo,angelo.marotta}@st.com

Abstract. We solve the manufacturing problem of identifying the model statistical parameters ensuring a satisfactory quality of analog circuits produced in a photolithographic process. We formalize it in a statistical framework as the problem of inverting the mapping from the population of the circuit production variables to the performances' population. Both variables and performances are random. From a sample of the joint population we want to identify the statistical features of the former producing a performance distribution that satisfies the design constraints with a good preassigned probability. The key idea of the solution method we propose consists of describing the above mapping in terms of a mixture of granular functions, where each is responsible for a fuzzy set within the input-output space, hence for a cluster therein. The way of synthesizing the whole space as a mixture of these clusters is learnt directly from the examples. As a result we have an analytical form both of the mapping approximating complex Spice models in terms of polynomials in the production variables, and of the distribution law of the induced performances that allows a relatively quick and easy management of the production variables' statistical parameters as a function of the probability with which we plan to satisfy the design constraint. We apply the method to case studies and real production data where our method outperforms current methods' running times and accuracies.

1 Introduction

A major challenge posed by new deep-submicron technologies is to design and verify integrated circuits to obtain a high fabrication yield, i.e. a high proportion of produced circuits that function properly. By contrast, with a further shrinking of process technology, the on-chip variation is getting worse for each technology node. Indeed, random and systematic defects as well as parametric process variations have a big influence on the yield of the manufactured circuits, with the consequence of frequent respinning of the whole development and manufacturing chain. This leads to high costs of multiple manufacturing runs and entails extremely high risks of missing a given market window. One way to overcome these drawbacks is to implement the DFM/DFY paradigm 

where Design for Manufacturability mates Design for Yield to form a synergistic manufacturing chain to be dealt with in terms of relationships between the statistical circuit parameters matching the production constraints and performance indicators ensuring correctly functioning dies. In line with previous works in the literature [23], we frame this problem in terms of random variable transforms, from the production variable space to the performance space, having the goal of identifying the best distribution law of the former inducing a satisfactory fulfillment of the performance requisites. The mapping from former to latter space is ruled by Spice models [4]. Our strategy is to manage it in terms of a granular construct based on a fuzzy partition of the production variable space. This allows a manageable interpolation of the Spice functions through polynomials and of the performances' joint distribution law as a mixture of the distribution laws affecting the single fuzzy sets. So we get a quick and accurate interpolation of the Spice functions and a satisfactory identification of the production statistical parameters via a method of moments that proves faster and more accurate than other procedures in the literature [53].

Our paper is organized as follows. In Section 2 we explain our procedure to identify the approximation of the Spice model and its adaptation to the experimental data. Since we base our inference on a method of moments, in Section 3 we toss the benefits and limits of this procedure on specific benchmarks in comparison with other methods. In the concluding section we outline a way of improving these results on the basis of the interpolated performances distribution *that is not constrained to be a Gaussian distribution* [6].

2 A Quick Model Identification

We formalize the modeling problem in terms of a mapping g from a random vector $\mathbf{X} = (X_1, \dots, X_n)$ ¹, describing what is commonly denoted as production variables, to a random vector $\mathbf{Y} = (Y_1, \dots, Y_t)$, describing what is commonly denoted as performances. The statistical features of \mathbf{X} , such as mean, variance, correlation, etc., constitute its parameter vector $\theta_{\mathbf{X}}$. Idem for the parameter vector $\theta_{\mathbf{Y}}$. Hence $\mathbf{Y} = g(\mathbf{X}) = (g_1(\mathbf{X}), \dots, g_t(\mathbf{X}))$, and we look for a vector $\theta_{\mathbf{X}}$ such that the corresponding $\theta_{\mathbf{Y}}$ characterizes a population where $P(\mathbf{Y} \in D_{\mathbf{Y}}) = \alpha$, having denoted with $D_{\mathbf{Y}}$ the domain the designers expect the circuit performances to belong to and with α a satisfactory probability value. We fulfill this task through an inference method based on the moments of the available sample. Actually, its implementation is a bit complex since the parameters concern \mathbf{X} whereas the moments are measured on \mathbf{Y} . Hence we need a comfortable function relating the \mathbf{Y} information to $\theta_{\mathbf{X}}$. Namely, first we interpolate g in terms of a polynomial in x , then we proceed with the inference.

2.1 Interpolating a Spice Model

The most common tool for modeling an analog circuit is represented by the Spice simulator [7]. It consists of a program which in input a textual description of the circuit

¹ By default, capital letters (such as X, Y) will denote random variables and small letters (x, y) their corresponding realizations; bold versions (\mathbf{X}, \mathbf{x}) of the above symbols apply to vectors of the objects represented by the symbols.

elements (transistors, resistors, capacitors, etc.) and their connections, translates this description into nonlinear differential equations to be solved using implicit integration methods, Newton's method and sparse matrix techniques. A general drawback of Spice – and circuit simulators in general – is the complexity of the transfer function it implements to relate physical parameters to performances which hampers intensive exploration of the performance landscape in search of optimal parameters. In our paper we bypass this handicap through a principled philosophy of considering the region D_x where we expect to set the production variables as an aggregate of fuzzy sets in various respects [8]. First of all we locally interpolate the Spice function g through a polynomial, hence a mixture of monomials that we associate to the single fuzzy sets. Many studies show this interpolation to be feasible, even in the restricted form of using posynomials, i.e. linear combination of monomials through only positive coefficients [9]. While this constraint is crucial for solving efficiently convex optimization problems such as geometric programming [10], for our purposes it is superfluous. The granular construct we formalize is the following.

Given a Spice function g mapping from x to y (the generic component of the performance vector y), we assume the domain $D_x \subseteq \mathbb{R}^n$ into which x ranges to be the support of c fuzzy sets $\{A_1, \dots, A_c\}$, each pivoting around a monomial m_k . We consider this monomial to be a local interpolator that fits g well in a surrounding of the A_k centroid. In synthesis, we have $g(x) \simeq \sum_{k=1}^c \mu_k(x)m_k(x)$, where $\mu_k(x)$ is the membership degree of x to A_k , whose value is in turn computed as a function of the quadratic shift $(g(x) - m_k(x))^2$.

On the one hand we have one fuzzy partition of D_x for each component of y . On the other hand we implement the construct with many simplifications, in order to match specific goals. Namely:

- since we look for a polynomial interpolation of g , we move from point membership functions to sets, to a monomial membership function to g , so that $g(x) \simeq \sum_{k=1}^c \mu_k m_k(x)$. In turn, μ_k is a *sui generis* membership degree, since it may assume also negative values (see next section);
- since for interpolation purposes we do not need $\mu_k(x)$, we identify the centroids directly with a hard clustering method based on the same quadratic shift.

The clustering perspective. We exploit two nice bonuses of our clustering instance. On the one hand the training set is constructed by us; thus it may be as large as we want, being constrained only by the computational time required to run a Spice model on a huge set of instances. Namely, we focus on a superset of D_x , so large as to be confident of really including the latter, and uniformly draw m points x_r which we pair with the $y_r = g(x_r)$ s computed by Spice. On the other hand, denoting $m_k(x) = \beta_k \prod_{j=1}^n x_j^{\alpha_{kj}}$, if we work with logarithmic scales, the shifts we consider for the single (say the i -th) component of y are the distances between $z_r = (\log x_r, \log y_r)$ and the hyperplane $h_k(z) = w_k \cdot z + b_k = 0$, with $w_k = \{\alpha_{k1}, \dots, \alpha_{kn}\}$ and $b_k = \log \beta_k$, constituting the centroid of A_k in an adaptive metric. Indeed, both w_k and b_k are learnt by the clustering algorithm aimed at minimizing the sum of the distances of the z_r s from the hyperplanes associated to the clusters they are assigned to. The procedure we implement is the following.

Algorithm 1. Adaptive c-means for Mosfet

For given Spice function g from $D_x \in \mathbb{R}^n$ to $D_y \in \mathbb{R}^t$

1. Initialization.
 - 1.1. Draw uniformly m points $\mathbf{x}_r \in D_x$ and form the training set $T = \{(\mathbf{x}_1, g(\mathbf{x}_1)), \dots, (\mathbf{x}_m, g(\mathbf{x}_m))\}$.
 2. Then, for each component y_i of $g(\mathbf{x})$
 - 2.1. Set the number c of clusters.
 - 2.2. Randomly draw a set of c hyperplanes $h_k(z) = \mathbf{w}_k \cdot z + b_k$ each lying in \mathbb{R}^{n+1} , with $z = (\log \mathbf{x}, \log y_i)$.
 - 2.3. For each \mathbf{z}_r :
 - 2.3.1. compute $d(\mathbf{z}_r, h_k(\mathbf{z}_r)) = \left(\frac{\mathbf{w}_k \cdot \mathbf{z}_r + b_k}{\|\mathbf{w}_k\|} \right)^2$ for each h_k ;
 - 2.3.2. assign \mathbf{z}_r to the cluster A_k affected by the minimum over \tilde{k} of $d(\mathbf{z}_r, h_{\tilde{k}}(\mathbf{z}_r))$;
 - 2.3.3. update \mathbf{w}_k and b_k along the gradient of $d(\mathbf{z}_r, h_k(\mathbf{z}_r))$.
-

In this way we obtain for instance the clustering of a training set drawn in the domains $D_x = \{\text{Vth}_0, \text{Tox}\}$ and $D_y = \{\text{Vth}, \text{Ron}\}$, where Vth_0 denotes the threshold voltage for large range devices when substrate-source voltage $V_{BS} = 0$, and Tox the gate oxide thickness of a totally dielectric isolated Bicmos whose performance is measured in terms of threshold voltage (Vth) and on-resistance (Ron). We consider separately 3 clusters in the spaces $\{\text{Vth}_0, \text{Tox}, \text{Vth}\}$ and $\{\text{Vth}_0, \text{Tox}, \text{Ron}\}$ in logarithmic scales. So the points are grouped around 3 planes in each space as shown in the first row of Fig. .

With the clustering procedure we essentially learn the exponents α_{kj} with which the \mathbf{x} components intervene in the various monomials, whereas the β_{ks} remain ancillary parameters. Indeed, to get the polynomial approximation of $g(\mathbf{x})$ we compute the mentioned *sui generis* memberships through a simple quadratic fitting, i.e. by solving w.r.t. the vector $\boldsymbol{\mu} = \{\mu_1, \dots, \mu_c\}$ the quadratic optimization problem:

$$\boldsymbol{\mu} = \arg \min_{\tilde{\boldsymbol{\mu}}} \sum_{r=1}^m \left(g(\mathbf{x}_r) - \sum_{k=1}^c \tilde{\mu}_k \prod_{j=1}^n x_{rj}^{\alpha_{kj}} \right)^2 \quad (1)$$

where x_{rj} denotes the j -th \mathbf{x} component of the r -th element of the training set T , and μ_k s override β_{ks} .

We deal with two approximation problems with this fitting: i) the adequacy of the polynomial form; and ii) the data granulation. The former is a monotonically increasing function of the number c of discovered clusters, with the trivial optimal solution of one cluster for each training set element and the obvious search for a breakeven point. The latter is a function of the sampled points. We may appreciate the adequacy of the sample size by evaluating the confidence regions around each monomial at a given confidence level. Since they read as hyperplanes in logarithmic scales, we may draw numerous methods in the literature for this task  .

To continue our previous example, in the second row of Fig.  we compare the accuracy of the polynomial approximation for increasing values of the number c of

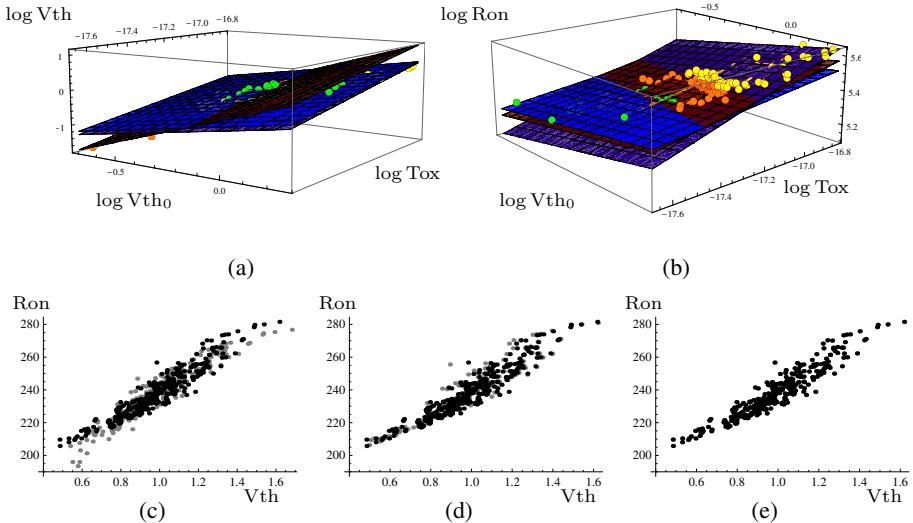


Fig. 1. The granular construct. First row: fuzzy clusters associated to output: (a) V_{th} , and (b) Ron , in logarithmic scales. Bullet color and radius are a function of cluster membership. Second row: polynomial approximation for $m = 300$ training points and $c = 3, 5$ and 10 , respectively. Black points: training set; gray points: its approximation.

monomials used for each y component, reckoning a percentage relative error of respectively 1.48%, 0.56% and 0.018%.

The statistical perspective. With the previous section we are left with a consolidated model for approximating \mathbf{g} , namely

$$y_i = \sum_{k=1}^c m_{ik}(\mathbf{x}) = \sum_{k=1}^c \mu_{ik} \prod_{j=1}^n x_j^{\alpha_{ikj}} \quad \text{for } i = 1, \dots, t \quad (2)$$

Now we use it to solve the inverse problem: Which statistical features of \mathbf{X} ensure a good (say 95%) coverage of the \mathbf{Y} domain where the designer assumes the circuit performances to fall in order to have a properly functioning circuit?

As an early solution of the problem we rely on the first and second moments of the target distribution, which are estimated on the basis of a sample S of sole \mathbf{Y} collected from the production lines as representatives of properly functioning circuits. Our goal is to identify the statistical parameters $\tilde{\theta}_{\mathbf{X}}$ of \mathbf{X} that produce through (2) a \mathbf{Y} population best approximating the above first and second order moments. The lead strategy is incremental, so that we numerically compute the involved moments of the monomials on the basis of the current instantiation of the candidate solution. This saves us from being hampered by the statistical complication deriving from an analytical computation of expected values of products of correlated variables.

In greater detail, \mathbf{X} is assumed to be a multidimensional Gaussian variable, so that we identify it completely through the mean vector $\nu_{\mathbf{X}}$ and the covariance matrix $\Sigma_{\mathbf{X}}$.

The analogous ν_Y and Σ_Y are a function of the former through (2) (but do not identify completely the \mathbf{Y} distribution). We use an alternating maximization strategy [13] that consists of identifying a $\tilde{\nu}_X$ approximation for fixed Σ_X and *vice versa*, until convergence. Namely, denoting with $\nu_{X_j}, \sigma_{X_j}, \sigma_{X_{j,k}}$ and $\rho_{X_{j,k}}$, respectively the mean and standard deviation of X_j and the covariance/correlation between X_j and X_k , using the Taylor series expansion limited to second order, we obtain an approximate expression of the sensitivity of ν_Y w.r.t. ν_X through

$$\frac{\partial \nu_{Y_i}}{\partial \nu_{X_j}} \simeq \sum_{k=1}^c \alpha_{ikj} \left(\frac{1}{\nu_{X_j}} + \frac{\sigma_{X_j}^2}{\nu_{X_j}^3} \right) \nu_{M_{ik}} \quad (3)$$

where M_{ik} on the right is a short notation of $m_{ik}(\mathbf{X})$, and its mean $\nu_{M_{ik}}$ is approximated with its sample estimate computed on samples generated with the current values of the parameters. Analogously, thanks to the approximations

$$\nu_{\Xi} \simeq \log \nu_X, \quad \sigma_{\Xi} \simeq \sigma_X / \nu_X, \quad \rho_{\Xi_{i,j}} \simeq \rho_{X_{i,j}} \quad (4)$$

with $\Xi = \log X$, coming from the Taylor expansion of respectively Ξ , $(\Xi - \nu_{\Xi})^2$ and $(\Xi_i - \nu_{\Xi_i})(\Xi_j - \nu_{\Xi_j})$ around (ν_{X_i}, ν_{X_j}) disregarding others than the second terms, and from the consequent rewriting of Σ_Y in terms of

$$\sigma_{Y_i}^2 = \sum_{k=1}^c \sigma_{M_{ik}}^2 + 2 \sum_{\substack{k,r=1 \\ k < r}}^c \sigma_{M_{ik,ir}}; \quad \sigma_{Y_{i,j}} = \sum_{k,r=1}^c \sigma_{M_{ik,jr}} \quad (5)$$

$$\text{with } \sigma_{M_{ik}}^2 \simeq \nu_{M_{ik}}^2 \left(\sum_{j=1}^n a_{ikj}^2 \frac{\sigma_{X_j}^2}{\nu_{X_j}^2} + 2 \sum_{\substack{j,r=1 \\ j < r}}^n \rho_{X_{j,r}} a_{ikj} a_{ikr} \frac{\sigma_{X_j} \sigma_{X_r}}{\nu_{X_j} \mu_{X_r}} \right) \quad (6)$$

$$\sigma_{M_{ik,ir}} \simeq \nu_{M_{ik}} \nu_{M_{ir}} \sum_{j,w=1}^n a_{ikj} a_{irw} \rho_{X_{j,w}} \frac{\sigma_{X_j} \sigma_{X_w}}{\nu_{X_j} \nu_{X_w}} \quad (7)$$

and analogous notation for σs in respect to their indexing by M and the double indexing of the latter, in turn, we numerically solve (5) in Σ_X when the left members coincide with the target value of Σ_Y .

3 Numerical Results

We test our method on the Spice model DIB12 of a bipolar NPN circuit. DIB technology achieves the full dielectric isolation of devices using SOI substrates by the integration of the dielectric trench that comes into contact with the buried oxide layer. Therefore, the variables commonly assumed to be both fluctuating as a function of the production technology and sensibly affecting the circuit performances are the following: ideal maximum forward beta (Bf), emitter resistance (Re), transport saturation current (Is),

and forward Early voltage (Vaf), which we rename as x_1, x_2, x_3 and x_4 , respectively. Whereas the remaining variables are set to standard values. In view of the typical values supplied by Spice, we assume the former to constitute a 4-dimensional Gaussian random vector \mathbf{X} with:

- mean $\boldsymbol{\nu}_\mathbf{X} = \{135, 0.8, 5.12 \times 10^{-18}, 138\}$,
- covariance matrix $\Sigma_\mathbf{X}$ with diagonal components $\sigma_\mathbf{X}$ equal to $0.05\nu_\mathbf{X}$, and correlations $\rho_{X_{i,j}}$ uniformly drawn in $[0.1, 1]$ for each i, j .

These features probabilistically define the circuit operational range D_x . From \mathbf{X} we draw three kinds of samples, respectively for: i) identifying the interpolating polynomial, ii) reconstructing parameters, and iii) computing run-time statistics. While for the first task a sample of 300 vectors \mathbf{x}_r is enough to get a good generalization (around same error with training and test set), we need a larger sample size (around 50 times the former) to support the numerical part of the estimation procedure. Actually, the quality of its results also depends on the size of the training set (the second of the above sample kinds), that must be large as well if we want to avoid some biases induced by the statistics.

With the former task, we build the \mathbf{Y} sample S made up of the corresponding $g_i(\mathbf{x}_r)$ s, with i indexing the output variables: current gain (Hfe), Early voltage (Early), and collector current (Ic). Of these variables we compute the following first and second order moment estimates:

$$\hat{\boldsymbol{\nu}}_\mathbf{Y} = \{110.57, 6.35 \times 10^{-5}, 111.803\}$$

$$\hat{\sigma}_\mathbf{Y} = \{5.72, 3.21 \times 10^{-6}, 5.91\}, \hat{\rho}_{Y_{1,2}} = 0.71, \hat{\rho}_{Y_{1,3}} = 0.46, \hat{\rho}_{Y_{2,3}} = 0.53$$

with the goal of inferring from them the original $\boldsymbol{\nu}_\mathbf{X}$ and $\Sigma_\mathbf{X}$. We implement an incremental procedure where at each run we update $\boldsymbol{\nu}_\mathbf{X}$ along the gradient descent on the quadratic error computed on $\hat{\boldsymbol{\nu}}_\mathbf{Y}$ using (3), and $\Sigma_\mathbf{X}$ as a solution of (5). The expected values $\nu_{M_{ik}}$ of the monomials are computed numerically on the basis of a large \mathbf{X} sample (third kind sample) generated with the current parameterization. In order to assess the robustness of the method, we start with notably perturbed values of $\boldsymbol{\nu}_\mathbf{X}$ and $\Sigma_\mathbf{X}$, respectively by a factor 1.5 and a random value between 0.5 and 1.5. Note that, since we have only 6 statistics available on \mathbf{Y} to identify 10 moments of \mathbf{X} , we decide to set 4 of them, namely $\tilde{\sigma}_{X_2}, \tilde{\rho}_{X_{1,4}}, \tilde{\rho}_{X_{2,4}}, \tilde{\rho}_{X_{3,4}}$ to the original values. The remaining parameters are inferred with a relative error of 0.13% and 1.4% respectively on first and second order statistics. The very good recovery of the \mathbf{Y} distribution law is shown in Fig. 2(a), denoting a percentage relative error between original and reconstructed points less than 0.23% over all coordinates, and a prediction error on the output parameters of 0.074% and 4.7% respectively on first and second order moments. However the method produces a positive bias on the variance estimates that may be visually perceived in the figure and requires a larger training set to be removed. On this regard, Fig. 2(b) refers to an experiment replica based on a training set of 10,000 elements. In this case we gain unbiased estimates with analogous prediction errors of 0.006% and 0.14%.

Coming to a true inference problem on the same circuit, we focus on manufacturing data made available by STMicroelectronics. They concern the same performances that

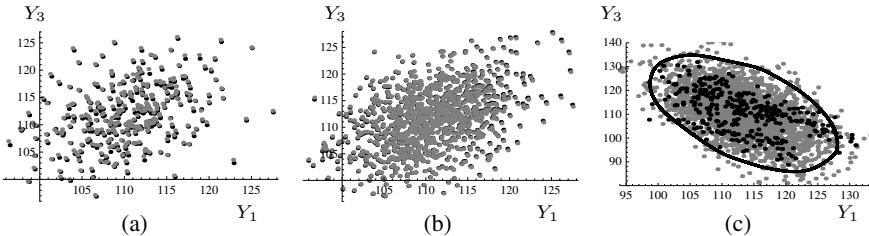


Fig. 2. Representing reconstruction and identification problems. (a-b) Comparison between output data and reconstruction provided by our procedure for training set size = 300 and 10,000, respectively. (c) Identification of a confidence region containing 95% of the experimental points (projected on the correlated components). Black bullets: original points; gray bullets: reconstructed points.

are collected through 322 (Y_1, Y_3) pairs and 161 Y_2 single values. From these data we compute the following statistics:

$$\begin{aligned}\hat{\nu}_Y &= \{113.197, 6.53 \times 10^{-5}, 110.594\} \\ \hat{\sigma}_Y &= \{6.89, 4.82 \times 10^{-6}, 11.24\}, \hat{\rho}_{Y_{1,3}} = -0.57.\end{aligned}$$

With these moments we again infer ν_X and Σ_X . Now we do not have original values for the parameters. So, to tackle the greater number of unknown parameters w.r.t. the available moments, we specialize the second part of the above procedure by alternating the inference of variances for fixed correlation (initially set equal to 0) and *vice versa*, until convergence. With this procedure we still are able to obtain a small relative error, less than 0.55% and 0.89% for first and second order output moments. However, we may expect a non perfect overlapping of empirical cumulative distribution functions (ECDFs) referred to the original performances and the reconstructed ones denoting some details of the distributions are missed by the new data because we rely just on first and second moments for their generation. In the concluding section we will sketch a way of recovering them with a more sophisticated modeling.

The operational goal of the questioned procedure is to ensure that a good percentage of the produced circuits satisfies the requests on performances issued by the designer. We may translate this goal in probabilistic terms by requiring that a good percentage, say 95%, of the produced population satisfies these constraints. In turn, we may identify the domain of such successful performances through a contour line of the Y distribution, namely a contour which includes the mentioned percentage, looking for a domain where 95% is also the percentage of sampled points falling inside it. Since we know the X parameters (because we inferred them) and also achieved a very good approximation of the Spice function through a polynomial, we generate a huge Y population with a cheap computation. Then we identify the desired contour line with a standard peeling method [14][15]. However, due to the well known limits of moment methods, we found a region that is too large, since it contains a percentage of sampled points equal to 97% rather than the set confidence. A first intervention we operate is a reduction of the variances (uniformly of a factor 0.90, in the absence of specific operational requirements)

so as to shrink the contour line. In this way we obtain the curve in Fig. 2(c) which contains exactly 95% of the experimental points.

In summary, with our approach we implement a procedure that compares with others in the literature w.r.t. the following standard indices: i) model building time: few seconds rather than the few minutes of most procedures considered in [5] and even hours with genetic algorithms [16]; ii) reconstruction error around four times less than other methods', which attests at around 1% [5]; and iii) prediction error less than around ten times than other methods (both on first and second order statistics) whose reference values are 1% and 10% respectively [3].

4 Conclusions

We solve a complex electronic manufacturing control problem using a granular construct. In spite of the methodology broadness the attribute *granular* may evoke, we obtain a very accurate solution benefitting from strict exploitation of state-of-the-art theoretical results, implemented through a fast algorithm as well. We also expect improvements with the full exploitation of the training set information content by elaborating our fuzzy set mixture directly in terms of a fuzzy mixture of the distributions affecting the single clusters. Since we are not drawn now by the need of a polynomial interpolating function, we may reintroduce the membership functions $\mu_k(x)$ of points to clusters and maintain the membership degrees of the monomial distributions to the \mathbf{Y} distribution as an extension of the *a priori* probabilities of the related clusters. We may compute the latter as a solution of the problem of minimizing the quadratic difference between the CDF deriving from a mixture of *corrected* density functions and the ECDF computed on the \mathbf{Y} sample S .

In Fig. 3(a) we sketch a way of succeeding in this regression with the same training set used to draw pictures in Fig. 1. Thanks to the analytical expression of the \mathbf{Y} CDF, we obtain a further refinement of the \mathbf{X} parameters based on the contour levels of the density function $f_{\mathbf{Y}}$. Namely, on the one hand we have the shape of the contour line containing, say, the 90% of the population, as shown in Fig. 3(b). We refine it both by centering it better w.r.t. the samples points – so that those falling outside the curve are

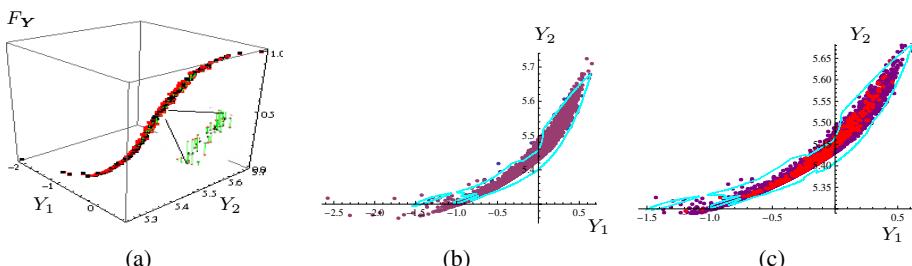


Fig. 3. Interpolating \mathbf{Y} distribution. (a) Comparison between CDFs. Black bullets: ECDF in the sample points; gray bullets CDF estimate in the same points; vertical lines: differences between the two values (in the zoomed portion). (b) Initial centering of the interpolated distribution. Bullets: simulated population; line: 90% contour line. (c) Final calibration. Same notation as in (b).

homogeneously distributed around it – and by tuning the \mathbf{X} parameters on the basis of the \mathbf{Y} CDF gradient, so that the new curve in Fig. 3(c) again represents a contour line with 0.90 coverage.

References

1. Bühler, M., Koehl, J., Bickford, J., Hibbeler, J., Schlichtmann, U., Sommer, R., Pronath, M., Ripp, A.: DFM/DFY design for manufacturability and yield - influence of process variations in digital, analog and mixed-signal circuit design. In: DATE 2006, pp. 387–392 (2006)
2. Qu, M., Styblinski, M.: Parameter extraction for statistical IC modeling based on recursive inverse approximation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 16, 1250–1259 (1997)
3. Koskinen, T., Cheung, P.: Statistical and behavioural modelling of analogue integrated circuits. Circuits, Devices and Systems, IEE Proceedings G 140, 171–176 (1993)
4. Quarles, T., Pederson, D., Newton, R., Sangiovanni-Vincentelli, A., Wayne, C.: Spice (2009), <http://bwrc.eecs.berkeley.edu/Classes/icbook/SPICE/>
5. McConaghy, T., Gielen, G.: Analysis of simulation-driven numerical performance modeling techniques for application to analog circuit optimization. In: Proceedings of IEEE International Symposium on Circuits and Systems (2005)
6. Bolt, M., Rocchi, M., Engel, J.: Realistic statistical worst-case simulations of VLSI circuits. IEEE Transactions on Semiconductor Manufacturing 4(3), 193–198 (1991)
7. Kundert, K.S.: The Designers Guide to SPICE and SPECTRE. Kluwer Academic Publishers, Boston (1998)
8. Apolloni, B., Bassis, S., Malchiodi, D., Witold, P.: The Puzzle of Granular Computing. Studies in Computational Intelligence, vol. 138. Springer, Heidelberg (2008)
9. Eeckelaert, T., Daems, W., Gielen, G., Sansen, W.: Generalized simulation-based posynomial model generation for analog integrated circuits. Analog Integr. Circuits Signal Process. 40, 193–203 (2004)
10. Hershenson, M., Boyd, S., Lee, T.: Optimal design of a CMOS op-amp via geometric programming. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 20, 1–21 (2001)
11. Apolloni, B., Bassis, S., Malchiodi, D., Pedrycz, W.: Interpolating support information granules. Neurocomputing 71, 2433–2445 (2008)
12. Sen, A., Srivastava, M.: Regression Analysis, Theory, Methods and Applications. Springer, Heidelberg (1990)
13. Gunawardana, A., Byrne, W.: Convergence theorems for generalized alternating minimization procedures. Journal of Machine Learning Research 6, 2049–2073 (2005)
14. Apolloni, B., Bassis, S., Gaito, S., Malchiodi, D.: Appreciation of medical treatments by learning underlying functions with good confidence. Current Pharmaceutical Design 13, 1545–1570 (2007)
15. Liu, R.Y., Parelius, J.M., Singh, K.: Multivariate analysis by data depth: Descriptive statistics, graphics and inference. The Annals of Statistics 27, 783–858 (1999)
16. McConaghy, T., Eeckelaert, T., Gielen, G.: CAFFEINE: Template-free symbolic model generation of analog circuits via canonical form functions and genetic programming. In: DATE 2005, pp. 1530–1591 (2005)

A New FGMOST Euclidean Distance Computational Circuit Based on Algebraic Mean of the Input Potentials

Cosmin Radu Popa

Faculty of Electronics, Telecommunications and Information Technology,
0011181 Bucharest, Romania
cosmin_popa@yahoo.com

Abstract. A new Euclidean distance circuit focused on high-speed operation will be presented in this paper. The computing accuracy will be improved compensating the error introduced by the second-order effects, which affect MOS transistor operation (short channel effect and mobility degradation) by a proper common-mode input voltage excitation of the squarer circuit. Because the elementary approach of designing an Euclidean distance circuit (exclusively based on classical MOS transistors in saturation) requires an additional threshold voltage extractor circuit, the new proposed idea is to use a FGMOST (Floating Gate MOS Transistor), having the advantage of a very large reducing of the circuit complexity.

Keywords: Computational circuits, FGMOST, VLSI design.

1 Introduction

Because of the square-law model of MOS transistor working in saturation, many analog signal-processing functions can be achieved using this square characteristic. Based on this principle, several basic building blocks, such as multipliers, active resistors and transconductors have been developed [1,2,3,4,5]. The Euclidean distance between $V_a = (V_{a1}, V_{a2}, \dots, V_{an})$ and $V_b = (V_{b1}, V_{b2}, \dots, V_{bn})$ (two n-dimensional vectors), defined as:

$$\|V_a - V_b\| = \sqrt{\sum_{k=1}^n (v_{ak} - v_{bk})^2}. \quad (1)$$

represents a direct measure of similarity between vectors V_a and V_b . The area of utilization of the Euclidean distance circuits includes instrumentation systems, communication circuits, neural networks, display systems [6] or several classification algorithms such as vector quantization or nearest neighbour classification. Several constraints (circuit area, power dissipation and modularity) limit the possibilities of implementing a high-speed Euclidian distance circuit. Choosing an analog VLSI implementation fulfills all the previous requirements, resulting the necessity of finding a simple and accurate circuit capable to compute the desired function, Euclidean distance.

2 Principle of Operation

The n-th order Euclidian distance circuit consists in n voltage squarer circuits and a square-root circuit.

2.1 The CMOS Square Circuit

First-order analysis. The proposed CMOS squarer using algebraic mean of the input potentials is presented in Fig. 1. Because $I_{D1} + I_{D2} = I_{D2} + I_{D3} = I_{D3} + I_{D4}$, it results that $I_{D1} = I_{D3}$ and $I_{D2} = I_{D4}$. Considering that all transistors from Fig. 1 are working in saturation and are characterized (in a first-order analysis) by a square characteristic, it is possible to write that $V_{GS1} = V_{GS3}$ and $V_{GS2} = V_{GS4}$, resulting a V potential equal with the algebraic mean of the input potentials:

$$V = \frac{V_a + V_b}{2}. \quad (2)$$

In this case, the output current of the circuit from Fig. 1 will have the following expression:

$$I_O = I_1 + I_2 - I = \frac{K}{2}(V_a - V_T)^2 + \frac{K}{2}(V_b - V_T)^2 - K\left(\frac{V_a + V_b}{2} - V_T\right)^2. \quad (3)$$

equivalent with:

$$I_O = \frac{K}{4}(V_a - V_b)^2 \quad (4)$$

So, the circuit computes the square of the differential input voltage.

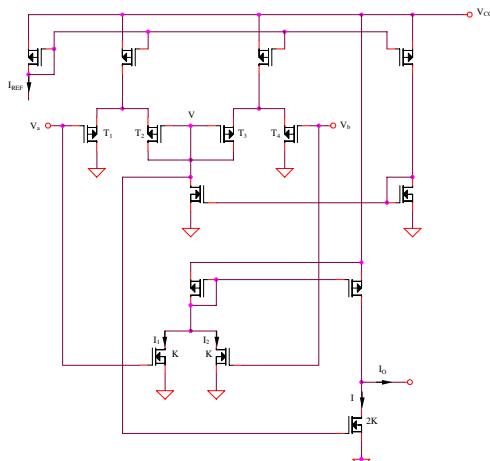


Fig. 1. CMOS squarer using algebraic mean of the input potentials

Second-order effects. Considering the second-order effects, which affect the MOS transistor operation, that is short-channel effect (5) and mobility degradation (6):

$$I_D = \frac{K}{2}(V_{GS} - V_T)^2(1 + \lambda V_{DS}) \quad (5)$$

$$K = \frac{K_0}{[1 + \theta_G(V_{GS} - V_T)](1 + \theta_D V_{DS})} \quad (6)$$

and supposing that the design condition $\lambda = \theta_D$ is fulfilled, it results that the gate-source voltage of a transistor working in saturation will have the following expression:

$$V_{GS} \cong V_T + \sqrt{\frac{2I}{k}} + \theta_G \frac{I}{k} \quad (7)$$

In this case, because $I_{D1} = I_{D3}$ and $I_{D2} = I_{D4}$, the relations $V_{GS_1} = V_{GS_3}$ and $V_{GS_2} = V_{GS_4}$ still remain valuable, so, even in a second-order analysis, the potential V will have the same expression (2). It results:

$$I_O = \frac{K}{4}(V_a - V_b)^2 - \theta_G[(V_a - V_T)^3 + (V_a - V_T)^3 - 2(\frac{V_a + V_b}{2} - V_T)^3]. \quad (8)$$

After algebraic calculations, the output current will have the following expression:

$$I_O = \frac{K}{4}(V_a - V_b)^2 - \frac{3}{4}\theta_G(V_a - V_b)^2(V_a + V_b - 2V_T) \quad (9)$$

The last term of (9) represents the error introduced by the mobility degradation. It is obvious that if the common mode input voltage is a little bit smaller than the MOS threshold voltage, that is if $V_a = v_a + V_T$ and $V_b = v_b + V_T$, with $v_a, v_b \ll V_T$, this error could be strongly reduced, resulting:

$$\epsilon_r = 3\theta_G(v_a + v_b) \ll 6\theta_G V_T \quad (10)$$

2.2 The CMOS Square-Root Circuit

Because of the square characteristic of MOS transistor working in saturation, a square-root circuit is relatively easy to implement in CMOS technology. The simplest approach is referring to a gate-source connected NMOS transistor (Fig. 2).

The circuit output voltage is:

$$V_O = V_T + \sqrt{\frac{2I_O}{k}} \quad (11)$$

The main disadvantage of the proposed square-root circuit is the necessity of using a threshold voltage extractor circuit to compute V_T , in order to remove the first undesired term from relation (11). As a result, the extreme simplicity of the proposed circuit will be considerably increased by this block. Three possible implementations of threshold voltage extractor circuits, preferred for theirs relatively small silicon area consumption due to the removing of any resistor from theirs design are presented in Fig. 3, 4 and 5, respectively.

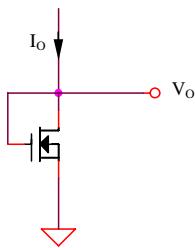


Fig. 2. The CMOS square-root circuit using a gate-source connected NMOS transistor

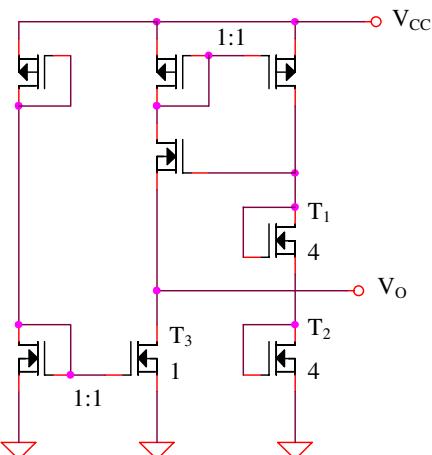


Fig. 3. Threshold voltage extractor - version I

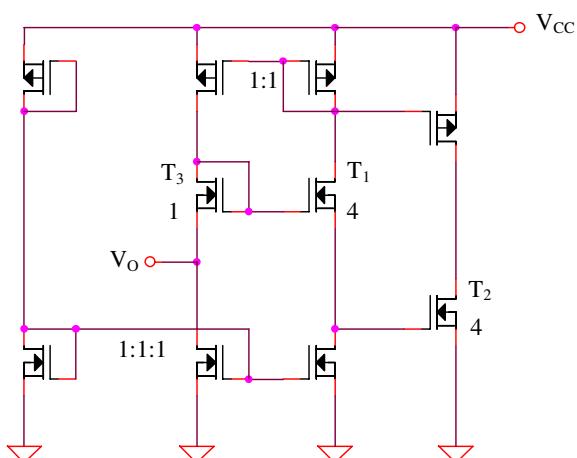


Fig. 4. Threshold voltage extractor - version II

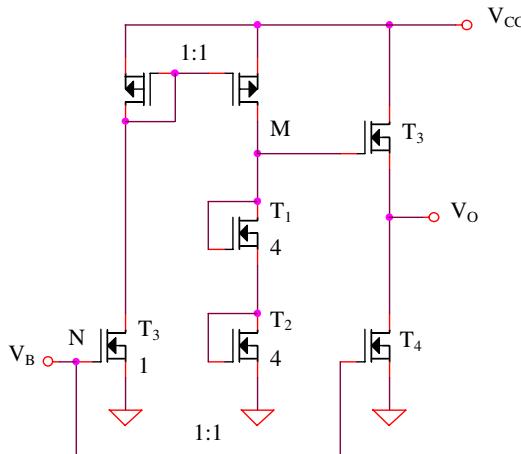


Fig. 5. Threshold voltage extractor - version III

The output voltage of the first two voltage extractor circuit could be written as:

$$V_O = V_{GS_1} + V_{GS_2} - V_{GS_3} \quad (12)$$

Considering that all the MOS transistors are working in saturation at the same drain current and supposing that the aspect ratio of T_1 and T_2 transistors are four time greater than the aspect ratio of T_3 , the output voltage will be equal with the threshold voltage of the NMOS devices:

$$V_O = V_{T_n} \quad (13)$$

For the same reason, the differential voltage $V_M - V_N$ (for the third V_T extractor circuit) will be equal with the threshold voltage. A difference circuit $T_3 - T_4$ (working in saturation) is used in order to refer the output voltage of the circuit to the ground. The output voltage of the difference circuit (which is the same with the output voltage of the threshold voltage extractor circuit) will have the following expression:

$$V_O = V_M - V_N = V_{T_n} \quad (14)$$

2.3 The CMOS Square-Root Circuit Using a FGMOSt (Floating Gate MOS Transistor)

The multiple-input floating-gate transistor is an ordinary MOS transistor whose gate is floating. The basic structure of a n-channel floating-gate MOS transistor is shown in Fig. 6a. The floating-gate is formed by the first silicon layer over the channel while the multiple input gates are formed by the second polysilicon layer, which is located over the floating-gate. This floating-gate is capacitive coupled to the multiple input gates. The symbolical representation of such devices is shown in Fig. 6b.

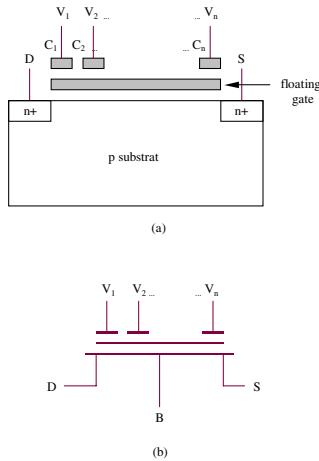


Fig. 6. (a) N-channel FGMOS transistor; (b) symbolic representation

The drain current of a FGMOS transistor with n-input gates in the saturation region is given by the following equation:

$$I_D = \frac{K}{2} \left[\sum_{i=1}^n k_i (V_i - V_S)^2 - V_T \right]. \quad (15)$$

where $K = \mu_n C_{ox} (W/L)$ is the transconductance parameter of the transistor, μ_n is the electron mobility, C_{ox} is the gate oxide capacitance, W/L is the transistor aspect ratio, $k_i, i = 1, \dots, n$ are the capacitive coupling ratios, V_i is the i-th input voltage, V_S is the source voltage and V_T is the threshold voltage of the transistor. The capacitive coupling ratio is defined as:

$$k_i = \frac{C_i}{\sum_{i=1}^n C_i + C_{GS}}. \quad (16)$$

where C_i are the input capacitances between the floating-gate and each of the i-th input and C_{GS} is the gate-source capacitance which is equal to $(2/3)C_{ox}$ for operation in the saturation region. All the overlap capacitances are assumed to be considerably smaller than capacitances summation $\sum_{i=1}^n C_i + C_{GS}$. Equation (15) shows that the FGMOS transistor drain current in saturation is proportional to the square of the weighted sum of the input signals, where the weight of each input signal is determined by the capacitive coupling ratio of the input. The proposed idea is to replace the classical NMOS transistor from Fig. 2 with a FGMOST for implementing the square-root circuit. The main goal of this replacement is the possibility of removing the first term from relation (11) by a proper polarization of the second gate of the FGMOST, that is without using any additional threshold extractor circuit and, in consequence, with an important reducing of the silicon area occupied by the Euclidean distance circuit. The FGMOS implementation of the square-root circuit is presented in Fig. 7.

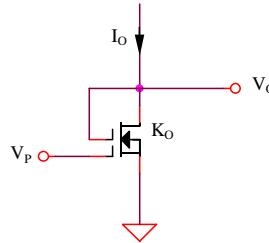


Fig. 7. The FGMOS square-root circuit

Applying relation (15) for the circuit presented in Fig. 7, it results that:

$$I_O = \frac{K_O}{2}(kV_O - kV_B - V_T)^2 \quad (17)$$

Considering a polarization voltage V_B that satisfies the condition $V_B = V_T/k$, the output voltage of the circuit from Fig. 7 will have the following expression:

$$V_O = \frac{1}{k} \sqrt{\frac{2I_O}{K_O}} \quad (18)$$

3 Euclidean Distance Circuit Using a FGMOSt

The block diagram of the FGMOSt Euclidean distance circuit is presented in Fig. 8. Considering n square circuits from Fig. 1 and the FGMOS square-root

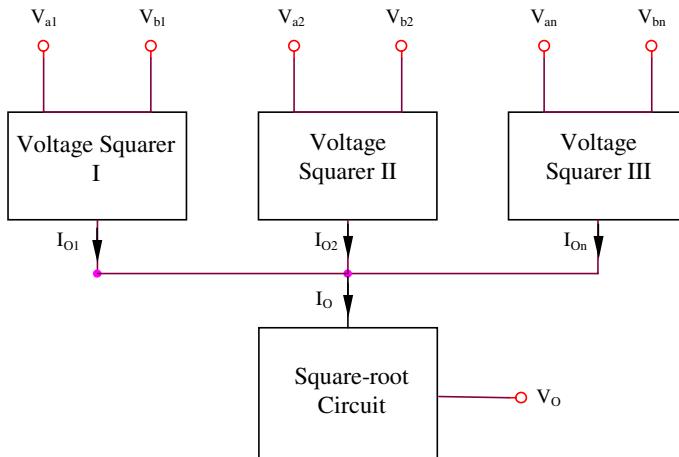


Fig. 8. The block diagram of the FGMOSt Euclidean distance circuit

circuit from Fig. 7 and imposing that $K/2K_o = k^2$, the output voltage of the Euclidean distance circuit will have the following expression:

$$V_O = \sqrt{\sum_{k=1}^n (v_{ak} - v_{bk})^2}. \quad (19)$$

4 Conclusions

The proposed circuit implements three important functions in the area of computational circuits: the squaring and square-root functions and the Euclidian distance. In order to reduce the circuit complexity by removing the necessity of using a threshold voltage extractor block, a FGMOS transistor is used. An original technique for reducing the squarer errors introduced by the second-order effects that affect the MOS transistor operation have been presented.

Acknowledgement

This work was supported by the Research Projects ID_916 and ID_1045 (financed by Romanian Ministry of Education, Research and Youth and UEFISCSU).

References

1. Bult, K., Wallinga, H.: A CMOS Four-Quadrant Analog Multiplier. *IEEE Journal of Solid-State Circuits*, 430–435 (1986)
2. Bult, K., Wallinga, H.: A Class of Analog CMOS Circuits Based on the Square-Law Characteristics of an MOS Transistor in Saturation. *IEEE Journal of Solid-State Circuits*, 357–364 (1987)
3. Seevinck, B., Wassenaar, R.F.: Versatile CMOS Linear Transconductor/Square-Law Function Circuit. *IEEE Journal of Solid-State Circuits*, 366–377 (1987)
4. Song, H.J., Kim, C.K.: A MOS Four-Quadrant Multiplier Using Two Input Squaring Circuits with Source Followers. *IEEE Journal of Solid-State Circuits*, 841–847 (1990)
5. Sakurai, S., Ismail, M.: A CMOS Square-Law Programmable Floating Resistor Independent on the Threshold Voltage. *IEEE Transactions on Circuits and Systems II*, 565–574 (1992)
6. Seevinck, B., Wassenaar, R.F., Wong, C.K.: A Wide-Band Technique for Vector Summation and Rms-Dc Conversion. *IEEE Journal of Solid-State Circuits*, 311–318 (1984)

FPGA Implementation of Support Vector Machines for 3D Object Identification

Marta Ruiz-Llata and Mar Yébenes-Calvino

Departamento de Tecnología Electrónica
Universidad Carlos III de Madrid
C/ Butarque 15, 28911, Leganés, Madrid
marta.ruiz-llata@uc3m.es

Abstract. In this paper we present a hardware architecture for a Support Vector Machine intended for vision applications to be implemented in a FPGA device. The architecture computes the contribution of each support vector in parallel without performing multiplications by using a CORDIC algorithm and a hardware-friendly kernel function. Additionally input images are not preprocessed for feature extraction as each image is treated as a point in a high dimensional space.

Keywords: Support Vector Machines, embedded systems, image processing.

1 Introduction

There has been an impressive development of cameras and distributed sensors that allows the positioning of many of them to monitor the environment. Application examples are ambient intelligence, intelligent transportation systems, security, and surveillance. However, to manage such an amount of information from a central computing system is very difficult, so there is a lot of interest in conceiving sensors that solely provide the pertinent information. A key issue is the development of adaptive embedded vision systems, because a great deal of information can be extracted from a single image or a video sequence, and vision tasks are very computer resource consuming.

Neural networks [1] and other learning machines algorithms, as Support Vector Machines (SVM) [2], have been successfully used in vision systems as they can deal with many image classification and pattern recognition tasks. However the implementation of these algorithms on limited resource hardware, such microprocessor or Field Programmable Gate Arrays (FPGAs) is still a very challenging task [3].

In this paper we propose the implementation of an adaptive vision system in FPGA devices. Neural processors for vision [4] and more general applications [5] have been implemented on FPGAs. These devices provide many advantages such solid development tools, easy reprogrammability and fast development time without losing performance with respect full-custom systems design. Additionally a high degree of parallel processing can be achieved compared with other

embedded platforms as microcontrollers and DSPs. The algorithm to be implemented in the FPGA device is the feedforward phase of a Support Vector Machine trained to identify different objects from the COIL data set [6] independently of their pose, as described in the software implementation of a SVM for 3D object recognition [2]. SVM have been also implemented on embedded hardware as FPGA [7] and microcontroller [8] with successfully results on classification error for reduced dimension input data. In reference [2], a SVM algorithm is proposed to perform recognition on images regarded as points in a high dimensional space, that is, the system does not require feature extraction so input images are treated as vectors. Our aim is also to process high dimension input vectors for object identification but in a dedicated hardware platform based on FPGA devices.

The structure of the paper is as follows. In the next section we describe the basic of SVMs. In section 3 we present the proposed hardware architecture of the feed forward phase of the SVM classifier. In section 4 we present its implementation for a reduced complexity problem using the Iris data base. Section 5 describes preliminary results using the COIL images data base. We present conclusions and further research in section 6.

2 Theoretical Overview of SVMs

Support Vector Machines, as opposed to Neural Networks, evolve from the theory to the implementation and experiments. Their theoretical foundations are statistical learning theory and structural risk minimization [9]. The non-linear two-class classification task of a training data set of l elements:

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\} \text{ where } \mathbf{x}_i \in R^n \text{ and } y = \pm 1$$

consists on resolving the following quadratic programming problem (CQP) with a series of associated restrictions:

$$\text{maximize} \quad -\frac{1}{2}\alpha^T \mathbf{Q}\alpha + \mathbf{r}^T \alpha \quad (1)$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, i = 1, \dots, l$$

$$\mathbf{y}^T \alpha = 0 .$$

where $r_i = 1 \forall i$, and \mathbf{Q} is a $l \times l$ positive semi-definite symmetric matrix and each element determined by:

$$q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) . \quad (2)$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function.

The feed-forward estimation or classification of a new, no learned, vector \mathbf{x} is:

$$y(\mathbf{x}) = \text{sgn} \left(\sum_{i=0}^l y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \right) . \quad (3)$$

where parameters α_i and b are given in the learning phase by resolving equation (1). The fact that the summation in equation (3) goes over all training data patterns is not relevant because most α_i parameters result equal to 0. Those training patterns whose corresponding α_i parameter is not equal to 0 are called support vectors (SV). The scarcity of the model results from a sophisticated learning that matches the model capacity to the data complexity ensuring a good performance on the future, previously unseen, data.

Typical Kernels are polynomial functions and Gaussian function. In addition other kernel function have been proposed in the literature; it is highlighted the function named hardware-friendly kernel function [10], which greatly simplifies the SVM feed-forward phase computation in resource constrained hardware while maintains good classification performance respect to the conventional Gaussian kernel [10]. Table II shows the examples of kernel functions.

Table 1. Kernel Functions

Kernel Funcions	
$[(\mathbf{x}^T \mathbf{x}_i) + 1]^d$	Polinomial of degree d
$e^{(-\ \mathbf{x}_i - \mathbf{x}\ ^2 / \sigma^2)}$	Gaussian
$2^{-\gamma \ \mathbf{x}_i - \mathbf{x}\ _1}$	Hardware-friendly

3 Proposed System Architecture

The goal is to map the classification algorithm of a Support Vector Machine represented by equation (3) into an FPGA. The basic architecture is represented in Fig. I

We compute the sign of the addition of $(n_{SV}+1)$ terms, being n_{SV} the number of support vectors, obtained from the training algorithm (equation (1)). Support vectors are those whose α_i parameter is not 0, so the summation in equation

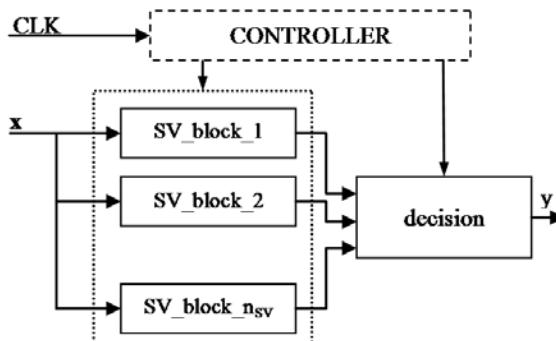


Fig. 1. Architecture for the calculation of the feed forward phase on an SVM classifier

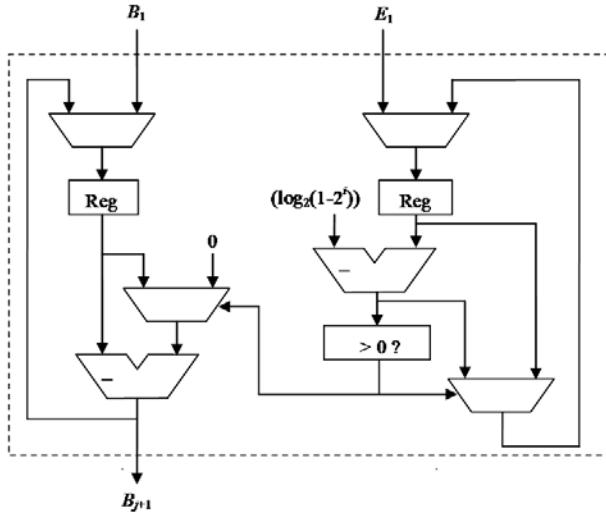


Fig. 2. Architecture for performing a single step of the hardware friendly kernel computation based on a CORDIC algorithm

(3) apply to the support vectors plus parameter b . The calculation of the terms for each support vector are performed in parallel by the blocks named SV_block. The additional block (Controller) is a state machine that controls the signal flow and the operation stages to compute the kernel function as described below.

The function of each SV_block is the computation of the hardware-friendly kernel represented in Table II as described in reference [10]. It is possible to compute the kernel function without resorting to any multiplication or function evaluation procedure by using a COrordinate Rotation DIgital Computer (CORDIC) algorithm. CORDIC algorithms are a class of hardware-efficient algorithms that provide iterative solutions based on shifts and adds for the calculation of trigonometric and transcendental functions [11].

As described in [10] the first step is to rescale the SVM learning parameters to guarantee their correct coding in the hardware. Their new values are between 0 and $1 - 2^{-k}$, being k the number of bits assigned to represent the parameters. The scaling procedure does not affect the classification function (equation (3)) as the sign remains unchanged. The evaluation of the kernel function is performed by the iterative algorithm described by equations (4) and (5):

$$B_{j+1} = B_j(1 + d_j 2^{-j}) . \quad (4)$$

$$E_{j+1} = E_j - \log_2(1 + d_j 2^{-j}) . \quad (5)$$

where $d_j = \{-1, 0\}$ is chosen such that $|E_{j+1}| \leq |E_j|$. Starting values are $B_1 = \alpha_i$ and for support vector \mathbf{x}_i and input vector \mathbf{x} . After k iterations (being k the number of bits assigned to represent the parameters) we obtain the solution:

$$B_{k+1} = B_1 2^{E_1} = \alpha_i 2^{-\gamma \|\mathbf{x}_i - \mathbf{x}\|_1} = \alpha_i K(\mathbf{x}_i, \mathbf{x}) . \quad (6)$$

The architecture for computing equations (4) and (5) of the CORDIC algorithm is represented in Fig. 2. It is based on the architectures proposed in references [10] and [11].

The additional elements that compound each SV_block are a memory which store k precomputed values of $\log_2(1 + d_j 2^{-j})$, a memory to store the α_i (B1) parameter and the support vector \mathbf{x}_i obtained from the training step, all represented with k bits. Finally an additional subblock (E1_block) has been included to perform the calculation of E_1 (see Fig. 2) from input vector \mathbf{x} and support vector \mathbf{x}_i . The architecture of SV_block is represented in Fig. 3. Here, E1_block is very important because it is designed in order to guarantee the convergence of the CORDIC algorithm. E_1 is restricted in the range (-1,0] thus its real value is divided into an integer part I and a decimal part F, so that the integral part is included at the end by shifting I times the output of the CORDIC algorithm.

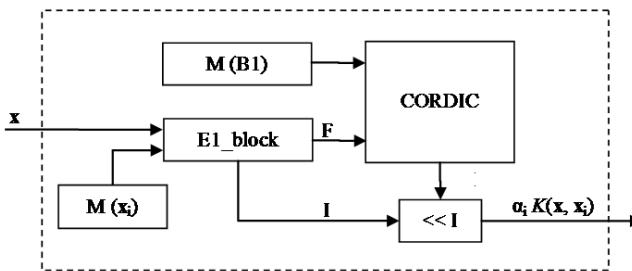


Fig. 3. Architecture for the calculation of the contribution of one support vector (SV_block)

4 Iris Data Set Implementation

We have implemented a two-class classifier for the simplified Iris dataset. The target device is a low cost Altera EP2C20 Cyclone II. This dataset is composed of 120 samples of two components vector that are non-linearly separable. The training step has been performed with Matlab software using the bioinformatics toolbox where the hardware-friendly kernel (Table II) has been included. Fig. 4 shows the training data set, the classification boundary and the resulting support vectors.

The classification hardware has been implemented with Altera Quartus II design software using schematic design and VHDL. The architecture is that represented in Fig. 1 just including as many SV_blocks as support vectors are provided in the training. The resolution chosen to represent vector components and SVM parameters is 8 bits, so the classification output is obtained after 10

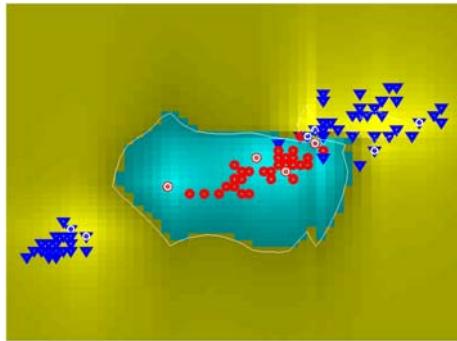


Fig. 4. SVM training and software testing

clock cycles, 8 consumed by the CORDIC algorithm. The maximum clock rate achieved, limited by E1_block, is 33 MHz, so each classification is performed in $0.3\mu s$. The performance of the SVM algorithm depends on the value of parameter γ in the hardware-friendly kernel function, the value of C in equation (1) and the number of elements randomly chosen in the training step. We used $C = 10$ and $\gamma = 2$. Good classifications results are obtained when about 10 support vector are selected in the training. In this case the classification error in the hardware SVM is below 10%, quite higher than the classification error performed by software simulation with the same number of support vectors (5%). This is due to low precision representation of the data in the hardware classifier, but it can be compensated by choosing a classifier with a higher number of support vectors. Increasing the number of support vector does not compromise classification speed as all support vectors are evaluated in parallel. Each SV_block occupies about 180 logic elements. The classifier with 10 support vectors takes 11% of logic cells of the target device and no additional hardware resources.

5 First Image Classification Results

The proposed hardware architecture is to be tested on the COIL database consisting of 7,200 images of 100 objects, there are 72 different angular views of each object. They are color images of 128×128 pixels. Reference [2] proposes a SVM for a multiclass classification of the COIL database using the images directly as inputs to the classifier. First, the images are converted to 8 bits levels gray-scale images and their resolution is reduced to 32×32 pixels, so they are represented as a vector with 1024 components. They obtain excellent recognition rates even with noise corrupted images, shifted images and partial occlusions. We will also follow the same strategy to deal with images as high dimensional input vector. Fig. 5 shows four examples of images to be classified.

In this case we also performed the training step with Matlab software using the bioinformatics toolbox where the hardware-friendly kernel has been included.

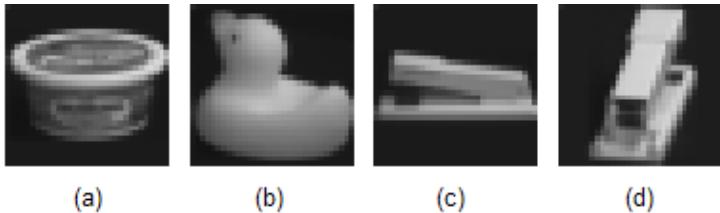


Fig. 5. Examples of modified images of the COIL dataset: (a) Obj72 “cream cheese” rotated 0° , (b) Obj74 “duck toy” rotated 45° , (c) Obj68 “stapler” rotated 0° and (d) Obj68 “stapler” rotated 100°

The classification stage is performed in software that simulates the hardware constrained architecture proposed in section 2. The classification results are compared with those provided by the Matlab toolbox programs.

First goal consists on adapting E1_block and the shift block (Fig.3) to deal with high dimensional input data (1024 vector elements). The first experiment consists on the classification of two object of the COIL database: for example the “duck toy” (obj74) and the “cheese cream” (obj72) (see Fig.5). Twelve images of each class corresponding to twelve different angles of view ($0^\circ, 30^\circ, 60^\circ, \dots$) have been used in the training. Learning parameters are $C = 10$ and $\gamma = 2$. After the training all the sample images are selected as support vectors, so the number of support vectors is 12. The other 60 images of each object have been used as test patterns. We achieved 0% classification error both in software SVM and FPGA hardware implementation. We have performed the same experiment with other image pairs from the COIL dataset. We observe that pair to pair comparison of images can be classified without errors when using a number of support vectors between 20 and 40. If we reduce the number of support vectors to 12, the classification error is kept below 5% in the hardware classifier. Worst cases correspond to images whose scale is changed depending on the angle of view as the “stapler” (see Fig.5 c and d). For example, the classification error of the hardware classifier between the “duck toy” and the “stapler” is 3.7%.

6 Conclusions

In this paper we have presented a FPGA implementation of a SVM based on a parallel computation of the support vectors kernel function. The starting point is the method proposed in [10], which provides a hardware-friendly kernel function that can be evaluated using an iterative CORDIC algorithm. Based on this method we have implemented the classification stage of a SVM on a low cost, constrained resources, FPGA device (Altera EP2C20 Cyclone II). The system shows to be very efficient in terms of hardware demanding resources and in terms of classification speed (each classification is performed in 10 clock cycles) compared with related implementations on other hardware platform as a micro-controller [8]. The system performance has been tested with the Iris database and a more sophisticated high dimensional image classification problem. Appealing

results have been obtained for low resolution (32×32) image classification without needing a preprocessing stage for feature extraction. Each image is treated as a point in a high dimensional space.

Next step will be to expand the architecture to a multiclass classification problem, being able to identify each object from all the others in the COIL dataset. For multiclass classification (q classes) we need to evaluate the procedure. As a first approach we will follow the same strategy as in reference [2] where it is necessary the training of $q(q - 1)/2$ between two classes and the evaluation of $q - 1$ classifiers in the hardware. We have estimated that in the target FPGA device up to 8 classifiers (with 12 support vectors each) can be evaluated in parallel. In this case of the image classification problem an external memory is needed to store the support vectors.

References

1. Malamas, E.N., Petrakis, E.G.M., Zervakis, M., Petit, L., Legat, J.: A survey on industrial vision systems, applications and tools. *Image and Vision Computing* 21, 171–188 (2003)
2. Pontil, M., Verri, A.: Support Vector Machines for 3D Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(6), 637–646 (1998)
3. Bouyssounouse, B., Sifakis, J. (eds.): *Embedded Systems Design*. LNCS, vol. 3436. Springer, Heidelberg (2005)
4. Yang, F., Paindavoine, M.: Implementation of an RBF Neural Network on Embedded Systems: Real-Time Face Tracking and Identity Verification. *IEEE Transactions on Neural Networks* 14(5), 1162–1175 (2003)
5. Lee, P., Costa, E., McBader, S., Clementel, L., Sartori, A.: LogTOTEM: A Logarithmic Neural Processor and its Implementation on an FPGA Fabric. In: IJCNN 2007, Orlando, FL (2007)
6. <http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php>
7. Anguita, D., Boni, A., Ridella, S.: A Digital Architecture for Support Vector Machines: Theory, Algorithm and FPGA Implementation. *IEEE Transactions on Neural Networks* 14(5), 993–1009 (2003)
8. Boni, A., Pianegiani, F., Petri, D.: Low-Power and Low-Cost Implementations of SVM for Smart Sensors. *IEEE Transactions on Instrumentation and Measurements* 56(1), 39–44 (2007)
9. Vapnik, V.N.: *The Nature of Statistical Learning Theory*. Springer, New York (1995)
10. Anguita, D., Pischiutta, S., Ridella, S., Sterpi, D.: Feed-Forward Support Vector Machine Without Multipliers. *IEEE Transactions on Neural Networks* 17(5), 1328–1331 (2006)
11. Andraka, R.: A survey of CORDIC algorithms for FPGA based computers. In: Proc. ACM/SIGDA 6th Int. Symp. Field Programmable Gate Array, Monterey, CA, pp. 191–200 (1998)

Reconfigurable MAC-Based Architecture for Parallel Hardware Implementation on FPGAs of Artificial Neural Networks Using Fractional Fixed Point Representation

Rodrigo Martins da Silva¹, Nadia Nedjah¹, and Luiza de Macedo Mourelle²

¹ Department of Electronics Engineering and Telecommunication

² Department of System Engineering and Computation,
Engineering Faculty, State University of Rio de Janeiro
`{rodrigo,nadia,ldmm}@eng.uerj.br`

Abstract. In this paper, we devise a hardware architecture for ANNs that takes advantage of the dedicated adder blocks, commonly called MACs, to compute both the weighted sum and the activation function. The proposed architecture requires a reduced silicon area considering the fact that the MACs come for free as these are FPGA's built-in cores. The implementation uses integer fixed point arithmetic and operates with fractions to represent real numbers. The hardware is fast because it is massively parallel. Besides, the proposed architecture can adjust itself on-the-fly to the user-defined configuration of the neural network, i.e., the number of layers and neurons per layer of the ANN can be settled with no extra hardware changes.

1 Introduction

Artificial Neural Networks (ANNs) are useful for learning, generalization, classification and forecasting problems [2]. They consist of a pool of relatively simple processing units, usually called artificial neurons, which communicates with one another through a large set of weighted connections. There are two main network topologies, which are feed-forward topology [2], [3], where the data flows from input to output units strictly forward, and recurrent topology, where feedback connections are allowed. Artificial neural networks offer an attractive model that allows one to solve hard problems from examples or patterns. However, the computational process behind this model is complex. It consists of massively parallel non-linear calculations. The software implementations of ANNs are useful but hardware implementations take advantage of the inherent parallelism of ANNs and so should answer faster.

Field Programmable Gate Arrays (FPGAs) provide a re-programmable hardware that allows one to implement ANNs rapidly and at low-cost. However, FPGAs lack the necessary circuit density as each artificial neuron of the network needs to perform a large number of multiplications and additions, which requires a lot of silicon area if implemented using standard digital techniques.

The proposed hardware architecture described throughout this paper is designed to process any fully connected feed-forward Multilayer Perceptrons (MLP) neural network. However, training is not included. It is now a common knowledge that the computation performed by the net is complex and consequently has a huge impact on the implementation efficiency and practicality. Existing hardware implementations of ANNs have attempted to speed up the computational process. However these designs require a considerable silicon area that makes them almost impossible to fit within the resources available on a state-of-the-art FPGAs [1], [4], [6]. In this paper, we devise an original hardware architecture for ANNs that takes advantage of the dedicated adder blocks, commonly called MACs (Multiply, Add and Accumulate blocks) to compute both the weighted sum and activation function.

The proposed implementation uses a specific number representation: Fractional Fixed Point [6]. It means that a real number is treated (approximated) by a fraction. Fractional addition, subtraction and multiplication are inherently integer (fixed point) operations, which may be an attractive choice in decreasing silicon area, because integer mathematics can be done by combinational circuitry.

The weighted sum of a neuron is now a sum of fractional products. In this project, the activation function (for all neurons of the ANN) is the sigmoidal logistic function, whose mathematics is also reduced (approximated) to additions, subtractions and multiplications of fractions. The exponential term $\exp()$ of the logsig function is approximated to 3 quadratic polynomials, using least-squares parabola method. The proposed architecture requires a reduced silicon area considering the fact that the MACs come for free as these are FPGAs built-in cores. The hardware is fast because it is massively parallel. Besides, the proposed hardware can adjust itself on-the-fly to the user-defined configuration of the neural network, with no extra hardware changes, which is a very nice characteristic in robot-like systems considering the possibility of the same hardware may be exploited in different tasks.

The remaining of this paper is organized as follows: In Section 2, we give a brief introduction to the computational model behind artificial neural networks; In Section 3, we show how we approximate the sigmoid output function, so we can implement the inherent computation using digital hardware; In Section 4, we provide some hardware implementation issues about the proposed design, that makes it original, efficient and compact; In Section 5, we present the detailed design of the proposed ANN Hardware; Last but no least important, In Section 6, we draw some useful conclusions and announce some orientations for future work.

2 ANNs Computational Model

We now give a brief introduction to the computational model used in neural networks. Generally, the net is constituted of few layers, each of which includes several neurons. The number of neurons in distinct layers may be different and consequently the number of inputs and that of outputs may be different [2].

The model of one artificial neuron requires m inputs, say x_1, x_2, \dots, x_m and the synaptic weights associated with these inputs, say w_1, w_2, \dots, w_m .

The weighted sum v , also called activation of the neuron, is defined in (1). The model usually includes an output (activation) function $\varphi()$ that is applied to the neuron activation before it is fed forwardly as input to the next layer neurons.

$$v = \sum_{i=0}^m x_j \times w_j \quad (1)$$

wherein $x_0 = 1$ and $w_0 = b$ (a noise in v) The non-linearity of the neuron is often achieved by the output function, which may be the hyperbolic tangent (*tansig*) or logistic sigmoid (*logsig*) [2]. In some cases, $\varphi(v)$ may be linear.

A typical ANN operates in two necessary stages: learning and *feed-forward computing*. The learning stage consists of supplying known patterns to the neural network so that the network can adjust the involved weights. Once the network has learned to recognize the provided patterns, the network is ready to operate, performing the feed-forward computing. In this stage, the network is supplied with an input data or pattern, which may be one of those used during the learning phase, and the corresponding output is then checked for correctness. This allows one to know whether the neural network could recognize the input data. The precision of the net in recognizing the new input patterns depends on the quality of its learning stage and on its generalization. As we have previously mentioned, here we are only concerned with the implementation of the feed-forward computing stage of the MLP Neural Network.

It is now suitable to study the numeric representation used in this project. We based our implementation on the Fractional Fixed Point (FFP) notation as described in [6]. The idea of this technique is to represent a real number as a fraction of integers. The main objective is to escape from representation and mathematic operations (addition, subtraction and multiplication) of the IEEE floating Point, which require considerable silicon area, in hardware terms. When a real number is treated as a fraction, only basic-integer operations (performed by combinational circuitry) are used, so that a more simple hardware is required.

Given a floating-point number a , e.g., somehow it is converted to a fraction using such an algorithm. This conversion is not a responsibility of the ANN hardware. The latter gets directly a fraction in a binary representation of 33 bits: numerator (16 bits) and denominator (17 bits). Numerator is a non-negative integer, i.e., natural. The arithmetic signal of the fraction is chosen arbitrarily to be placed in the bit length of denominator. So, this one has 16 bits to allocate a non-zero-natural number and more 1 bit to express the signal of the fraction (Fig. 1). Thus, we can also guarantee that the denominator is a not zero, wherein a is a floating-point number, N_a is the numerator (natural) and D_a is the denominator (non-zero integer).

$$\text{conversion: } a \longmapsto \frac{N_a}{D_a} \text{ or } a \longmapsto \text{frac}(a)$$

Any addition, subtraction or multiplication using fractions results in another fraction which may be considered as a natural numerator and a non-zero-integer denominator. Considering b another real number, below we see basic operations

with fractions. ANN hardware operates with a fraction in the binary representation of Fig. 1.

$$\begin{aligned} \text{addition: } a + b &\longmapsto \frac{N_a}{D_a} + \frac{N_b}{D_b} = \frac{N_a \times D_b + D_a \times N_b}{D_a \times D_b} \\ \text{subtraction: } a - b &\longmapsto \frac{N_a}{D_a} - \frac{N_b}{D_b} = \frac{N_a \times D_b - D_a \times N_b}{D_a \times D_b} \\ \text{multiplication: } a \times b &\longmapsto \frac{N_a}{D_a} \times \frac{N_b}{D_b} = \frac{N_a \times N_b}{D_a \times D_b} \end{aligned}$$

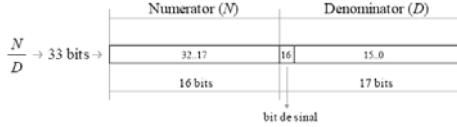


Fig. 1. Binary representation of a fraction

When a real number is converted into a fraction (considering a limited bit length for both numerator and denominator), there is a loss of accuracy involved. Therefore, it is quite important (in a conversion process) to search for a fraction that presents the minimum error regarding the given real number.

3 Approximation of the Output Function

Unlike the weighted sum, v , which includes operations that can easily and efficiently implemented in hardware, the activation or output function of a neuron requires a special care before the computation involved can be modeled in hardware. Without loss of generality, we chose to use the logistic sigmoid, defined in (2), as the output function for all neurons in the ANN hardware. To allow an efficient implementation of (2), in hardware, we proceeded with least-squares parabola method, using 3 quadratic polynomials in order to approximate the exponential decay: $\exp(-v)$. Initially we consider $v \geq 0$, as depicted in see Fig. 2. The same method could be applied to the tangent hyperbolic function too.

$$\varphi(v) = \text{sigmoid}(v) = \frac{1}{1 + e^{-v}} \quad (2)$$

In Fig. 2-(b), for example, the approximation method generates the following quadratic polynomial, for $v \in [0, 2]$:

$$\exp(-v) \approx f_{[0,2]}(v) = 0.1987234 \times v^2 - 0.8072780 \times v + 0.9748092 \quad (3)$$

We know the ANN hardware operates with fractions, whose representation is depicted in Fig. 1. So, it will be better to express the polynomial of (3) in the hardware parlance.

$$f_{[0,2]}(v) \approx \frac{12858}{64703} \times \left(\frac{N_v}{D_v}\right)^2 + \frac{11691}{-14482} \times \frac{N_v}{D_v} + \frac{56072}{57521} \quad (4)$$

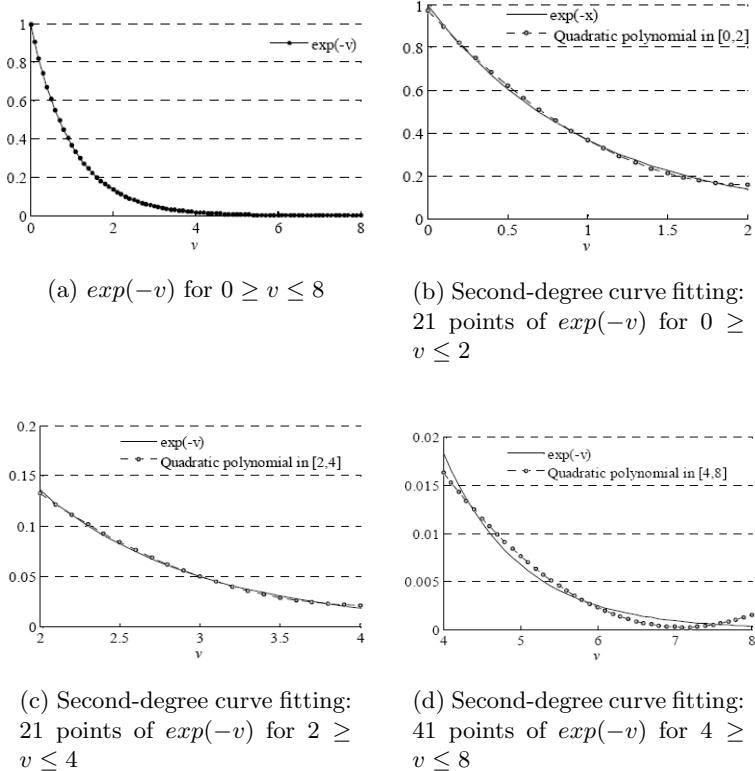


Fig. 2. Approximation of the exponential decay: $\exp(-v)$

In (4), there are 3 multiplications and 2 additions with fractions to be performed. As we are also concerned in time computation, it is possible to save one operation (multiplication) using the identity $a \times v^2 + b \times v = v(a \times v + b)$. Computation of the fraction polynomial in (4) leads to the fraction in (5), wherein $F_{[0,2]}(v)$ is a fractional function in $v \in [0, 2]$.

$$f_{[0,2]}(v) \approx F_{[0,2]}(v) = \frac{N_{F_{[0,2]}(v)}}{D_{F_{[0,2]}(v)}} \quad (5)$$

In general, for $v = 0$, we can say that $\exp(-v)$ is approximated by a fraction $F(v)$, so that $\exp(v) \approx F(v)$, wherein $F(v)$ can be $F_{[0,2]}(v)$, $F_{[2,4]}(v)$, $F_{[4,8]}(v)$ or $\text{frac}(0)$. Therewith, the ANN hardware computes an approximate-quadratic polynomial in a fractional form, when $v \in [0, 8]$; and for $v \geq 0$, the hardware approximate $\exp(-v)$ to zero, i.e., to $\text{frac}(0)$. Thus, the fractional form of the logistic sigmoid, $\varphi(v)$, defined in (2), is obtained using the considerations in (6) and (7).

$$\exp(v) \approx F(v) = \frac{N_{F(v)}}{D_{F(v)}} \Rightarrow \varphi(v) = \frac{1}{1 + e^{-v}} \approx \frac{1}{1 + F(v)} \quad (6)$$

$$\varphi(v) = \frac{1}{1 + \frac{N_{F(v)}}{D_{F(v)}}} \Rightarrow \varphi(v) \approx \frac{D_{F(v)}}{D_{F(v)} + N_{F(v)}}, \text{ wherein } v \geq 0 \quad (7)$$

The fractional form of $\varphi(v)$, for $v \geq 0$, is computed as follows:

$$\text{For } \frac{N_v}{D_v} < 0, \text{frac}(\varphi(v)) = \frac{N_{\varphi(v)}}{D_{\varphi(v)}} \text{ is easily obtained from } \frac{D_{F(v)}}{D_{F(v)} + N_{F(v)}} \quad (8)$$

When v is negative ($v < 0$), we can take advantage of a logistic-sigmoid property: $\varphi(v) = 1/\varphi(-v)$. Since $v > 0$, there is no problem if we use (6) and (7) to determine $\varphi(v)$, by replacing v by $-v$.

$$\varphi(-v) = \frac{1}{1 + \frac{N_{F(-v)}}{D_{F(-v)}}} \Rightarrow \varphi(-v) \approx \frac{D_{F(-v)}}{D_{F(-v)} + N_{F(-v)}}, \text{ wherein } -v \geq 0 \quad (9)$$

So,

$$\varphi(v) \approx 1 - \frac{D_{F(-v)}}{D_{F(-v)} + N_{F(-v)}} \Rightarrow \varphi(v) \approx \frac{N_{F(-v)}}{D_{F(-v)} + N_{F(-v)}}, \text{ wherein } v \leq 0 \quad (10)$$

Finally, for $v < 0$, the fractional form of $\varphi(v)$ is computed as follows:

$$\text{For } \frac{N_v}{D_v} < 0, \text{frac}(\varphi(v)) = \frac{N_{\varphi(v)}}{D_{\varphi(v)}} \text{ is easily obtained from } \frac{D_{F(-v)}}{D_{F(-v)} + N_{F(-v)}} \quad (11)$$

The intervals $[0,2]$, $[2,4]$ and $[4,8]$ were not chosen arbitrarily: the extremes 2, 4 and 8 are attractive when we evaluate if a fraction belongs to one of those intervals. For example, if $v \geq 0$, and so $\text{frac}(v) \geq 0$, the comparison in (12) is easily performed, in hardware, by left shifting and comparing (logical test). If, instead of 2 in (12), it were 4, hardware would shift twice, and so if it were 8, three times.

$$\frac{N_v}{D_v} < 2 \iff \frac{N_v}{2} < D_v \quad (12)$$

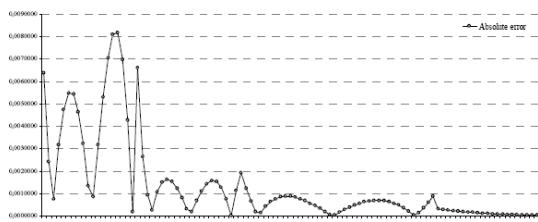


Fig. 3. Absolute Error: $|\varphi(v)\varphi_q(v)|$

In Fig. 3, we see the absolute error between $\varphi(v)$, calculated with no approximation, and $\varphi_q(v)$, which is obtained as an approximation of $\exp(v)$ by one of the approximate-quadratic polynomials (Fig. 2), for $v \in [0, 8]$; and if $v \geq 8$, $\exp(v)$ is approximated to 0 (zero). The maximum error obtained is ≈ 0.0081757 , in the interval $v \in [0, 2]$. We can see that the least-square-parabola approximation is better as curve-fitting in $v \in [2, 8]$.

4 Implementation Issues

An Artificial Neural Network is a set of several interconnected neurons arranged in layers. Let L be the number of layers. Each layer has its own number of neurons. Let m_i be the number of neurons in layer i . The neurons are connected by the synaptic connections. Some neurons get the input data of the network, so they are called input neurons and thus compose the input layer ($i = 1$). Other neurons export their outputs to the outside world, so these are called output neurons ($i = L$) and thus compose the output layer. Neurons placed on the layer $i = 2$ up to layer $i = L - 1$ are called the hidden neurons because they belong to the hidden layers.

The computation corresponding to a given layer starts only when that of the corresponding previous layer has finished. Our ANN hardware has just one real layer of neurons, which constitutes of n neurons, wherein n is maximum number of neurons per layer, considering all layers of the Neural Network to be computed. Besides reducing the number of neurons that are actually implemented in hardware, our design takes advantage of some built-in cores that come for free in nowadays FPGAs. These blocks are called MACs (Multiply, add and Accumulate), which are usually used in DSPs (Digital Signal Processing).

5 ANN Hardware Architecture

The ANN Hardware interface is illustrated in Fig. 4, wherein three other components are included: NNALU (Neural Network Arithmetic and Logic Unit), Control Unit of NNALU and Clock Generator. An external software is necessary to control ANN hardware computation. This software feeds the hardware (via Data Bus for Hardware/Software communication – DBHS) with the inputs, weights and biases required by the net. The software still defines (via Control Bus for Hardware/Software communication – CBHS) how many neurons per layer hardware is going to work and all necessary steps for accomplishing the whole feed-forward computing of the network.

Focusing on the ANN Hardware, Control Unit of NNALU encompasses all control components for computing all neural network specific operations: sums, products and comparisons with fractions. The blocks that perform these operations are within NNALU, which is the Arithmetic Logic Unit of the ANN Hardware. Clock Generator defines the time basis of the ANN Hardware and synchronizes all operations in NNALU (which are commanded by the Control Unit).

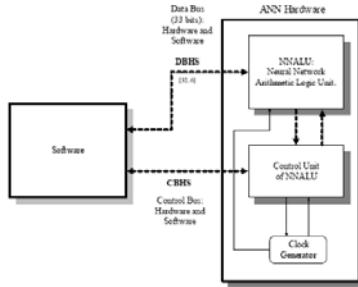


Fig. 4. Interface of the ANN hardware

Fig. 5-(a) shows details of NNALU. It is possible to see the physical layer with n Neurons. During the ANN Hardware operation, the Neurons work in parallel, even though not necessarily all physical neurons are needed in the layer. Recall that some layers in the Neural Network may have fewer neurons than n . As we can see, the output of each hardware neuron (Y_i) goes out of the NNALU, to furnish the output of the Network, when the whole computation is finished. Signal Y_i also returns to the Neurons by a tri-state buffer, a multiplexer MuxX and a register RegX. This Feedback action allows the reuse of one physical layer for computing all layers of the Neural Network (virtual layers). MuxX is also an interface of communication between the software and NNALU, wherein the latter receives *Inputs* and *Biases* from the software. MuxWi, on the other hand, allows the weights (that comes from the software) get into Neurons of the NNALU.

All Neurons in Fig. 5-(a) have the same hardware architecture, which is depicted in Fig. 5-(b). The latter shows Multiplier and Adder, which are used for the operations between numerators and denominators of fractions (Section 2). Still in Fig. 5-(b), the buses X , W , F and Y carry fractions in the model of Fig. 1. Registers are useful to make data stable during multiplications, additions or any other operation. Mux1 allows to select an operand for the Multiplier: a numerator or a denominator of X . Mux2 do the similar task, regarding W . This way, it becomes feasible to perform $X \times Y$, which are fractions.

The arithmetic signals of X and W (Fig. 5-(b)) go to ASPU (Fig. 6), where is given the signal of the fraction obtained by multiplying and adding two other fractions. ASPU also decides if an Adder operand will experiment the two's complement (Twocomple1 or Twocomple2, Fig. 5-(b)). The Adder component is responsible to additions of the neuron's weighted sum. LShfReg3 and Reg4 are assigned to accumulate the weighted sum as a fraction: numerator in LShfReg3 and denominator in Reg4.

In Fig. 5-(b), Comparator performs the logical test in (10). Numerator is left-shifted by LShfReg1 and compared with the denominator, which comes from LShfReg2. Once Hardware computes the output function (*logsig*) using an approximate-quadratic polynomial, the same components used in the weighted sum are also taken advantage, because a polynomial involves sums and products.

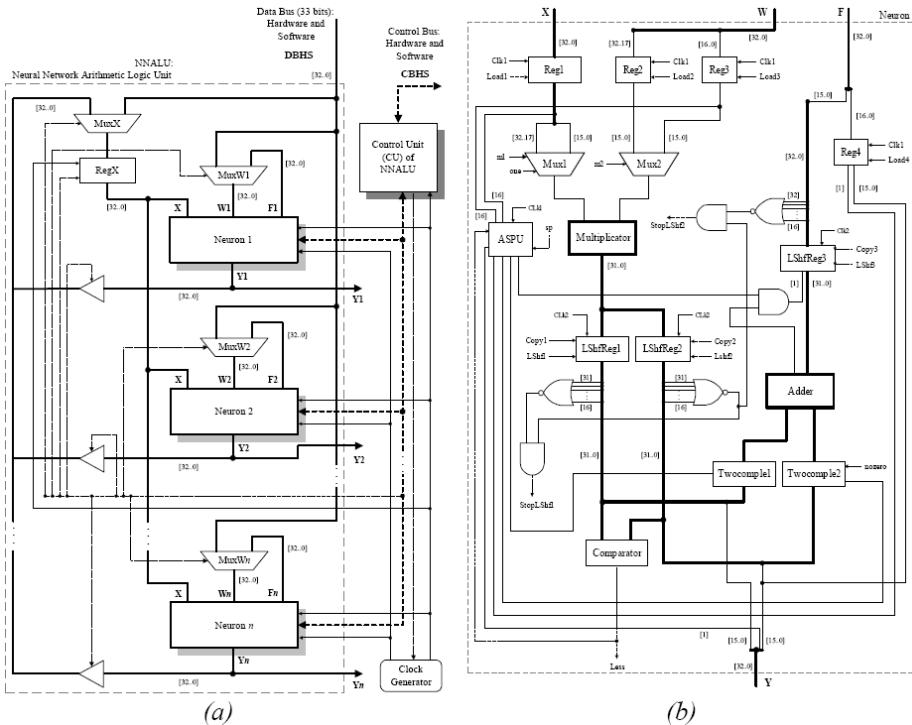


Fig. 5. Arithmetic Logic Unit and Neuron Architecture

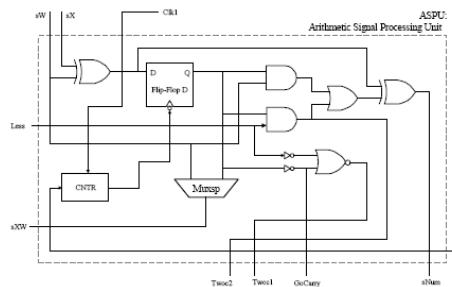


Fig. 6. Arithmetic Signal Processing Unit (fraction signal processing)

All Left Shifters (LShfReg1, ...) are responsible to keep numerator or denominator with 16 bits, each one, not considering the bit for arithmetic signal. For instance, if a multiplication is performed with two fractions, the result is a numerator and a denominator with 32 bits each one; then, left shifters are used to put them back with 16 bits, each one.

In Fig. 6, component CNTR releases the negative transition of Clk1 to the Flip-Flop D, if sp = 1. ASPU works in parallel with multiplications and additions

of fractions during weighted sum and output function computation. Each Neuron, in Fig. 5-(a), works in parallel with another one. If one layer is in computation, the next one is being prepared to initialize. When all layers is completely ended, ANN Hardware returns a signal to its Control Unit, informing that the output of the Neural Network is available. So, Control Unit returns to the software that the whole computation is done.

6 Conclusions

In this paper, we presented novel hardware architecture for processing an artificial neural network, whose configuration (number of neurons per layer and etc) can be changed on-the-fly without any extra effort. The design takes advantage of the built-in MACs block that come for free in modern FPGAs. The model was specified in VHDL, simulated to validate its functionality. We are now working on the synthesis process to evaluate time and area requirements. The comparison of the performance result of our design will be then compared to both the binary-radix straight forward design and the stochastic computing based design.

Acknowledgments

We are grateful to FAPERJ and CNPq for their continuous financial support.

References

1. Brown, B.D., Card, H.C.: Stochastic Neural Computation II: Soft Competitive Learning. *IEEE Transactions on Computers* 50(9), 906–920 (2001)
2. Hassoun, M.H.: *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge (1995)
3. Moerland, P., Fiesler, E.: Neural Network Adaptation to Hardware Implementations. In: Fiesler, E., Beale, R. (eds.) *Handbook of Neural Computation*. Oxford, New York (1996)
4. Nedjah, N., Mourelle, L.M.: Reconfigurable Hardware for Neural Networks: Binary radix vs. Stochastic. *Journal of Neural Computing and Applications* 72(10-12), 249–155 (2007)
5. Nedjah, N., Silva, R.M., Mourelle, L.M.: Dynamic MAC-based Architecture of Artificial Neural Networks Suitable for Hardware Implementation on FPGAs. *Neurocomputing* 16(3), 2171–2179 (2009)
6. Saint-Jones, P., Gu, D.: Fractional Fixed Point Neural Network: An introduction, Technical Report, Department of Computer Science, University of Essex, Wivenhoe Park, Colchester Essex

A Two Stage Clustering Method Combining Self-Organizing Maps and Ant K-Means

Jefferson R. Souza, Teresa B. Ludermir, and Leandro M. Almeida

Center of Informatics, Federal University of Pernambuco

Av. Prof. Luis Freire, s/n, Cidade Universitária Recife/PE, 50732-970, Brazil

{jrs2,tbl,lma3}@cin.ufpe.br

Abstract. This paper proposes a clustering method SOMAK, which is composed by Self-Organizing Maps (SOM) followed by the Ant K-means (AK) algorithm. SOM is an Artificial Neural Network (ANN), which has one of its characteristics, the nonlinear projection from a high dimensionality of the sensorial space. AK is based in the Ant Colony Optimization (ACO), which is a recently proposed meta-heuristic approach for solving hard combinatorial optimization problems. The AK algorithm modifies the K-means on locating the objects and these are then clustered according to the probabilities which in turn are updated by the pheromone. The SOMAK has a good performance when compared with some clustering techniques and reduces the computational time.

Keywords: Self-Organizing Maps, Ant Colony Optimization and Unsupervised Learning.

1 Introduction

With the substantial reduction of data storage cost, a great improvement in the performance of computers and the popularization of computer nets, a great amount of data information is being produced every day everywhere. So, a great quantity scale of databases has created the necessity of developing some techniques of data processing useful for the clustering of data or data mining [4].

The K-means algorithm is the most commonly used partitive clustering algorithm because it can be easily implemented and it is very efficient in terms of the execution time. The major problem with K-means is the definition of k to the clustering problem. SOM [6] is an ANN which allows the visualization of high dimensionality data and also implements an ordered mapping of a distribution of high dimension within a regular grid of low dimension. This ordered grid can be used as a convenient visualization to show different characteristics of the SOM. The algorithm SOMK, that is SOM followed by K-means, does not need to define the ideal number of k-clusters [12]. The algorithm AK is based on the Ant Colony Optimization (ACO) [3], recently proposed for solving hard combinatorial optimization problems. We used AK to find the optimized k [7]. The advantage of the proposed algorithm SOMAK is that it needs less time with small number of clusters to solve a problem. We have to point out that the aim of

this paper is not to find an optimal clustering for the data, but to obtain a view about the structure of data clusters using SOMAK, besides trying to reduce the number of clusters and the computational time.

In the case of the experiments, a comparison between the results of direct data (SOM and K-means) and the clusters of prototypes vectors of SOMK and SOMAK is carried out. This paper contains eight sections. Section 1 describes our research motivation. Section 2 contains the related work. Section 3 contains the methods of clustering. In section 4, a two-stage method combining SOM and Ant K-means is described. Section 5 shows the material and methods. Section 6 describes the experimental results and discussion. Section 7 contains the conclusion and future work. Finally, the acknowledgement are presented.

2 Related Works

Kuo et al. used AK in the analysis of clusters [7]. The algorithm AK modifies the k-means locating the objects and these are then clustered according to the probabilities which in turn are updated by the pheromone according to the total within cluster variance (TWCV). The experimental results showed that AK is better than the other two methods, SOMK and SOM followed by the genetic k-means algorithm [7]. The only problem for AK is that the number of clusters is required, that is, it is necessary to give the number of clusters to algorithm AK for it to be started.

Vesanto and Alhoniemi combined SOM and K-means [12] to solve the clustering problem. Particularly, the use of hierarchical agglomerative clustering and the partitive clustering using K-means are investigated. The procedure consists of two stages, firstly using a SOM to produce the prototypes, which are then clustered in the second stage by the K-means. The results of the clustering using a SOM as an intermediary phase was computationally effective, besides comparing the results directly obtained from the data, considering the original difficulties from the properties of the K-means algorithm. Trying to solve the needs of the algorithms which were seen and described above, we need to develop some useful techniques of data processing to improve the solution of the data clustering or data mining. So, this paper proposes a method of clustering based on two stages combining SOM and Ant K-means for the analysis of clusters.

3 Methods of Clustering

3.1 K-Means

The K-means method of clustering is one of the simplest algorithms of unsupervised learning to solve the clustering problem. The aim is to divide the data set within k clusters fixed a priori. The algorithm consists of two stages: an initial stage and an iterative stage. The initial stage involves the definition of the k centroids, one for each cluster. The second iterative stage repeats the signature of

each point of data for the closest centroid and k new centroids are calculated according to the new signature [10]. This interaction stops when a certain criterion is found; for example, number of interactions. Given a set $nPat$, suppose we want to classify the data within k groups, the algorithm tends to minimize a function of error, such as a mean squared error defined as: $E = \sum_{k=1}^C \sum_{i=1}^{nPat} \|x_i - c_k\|^2$. Where C represents the number of clusters, $nPat$ the number of samples, x the entry of each sample and c_k is the center of cluster k .

3.2 Ant Colony Optimization

The ACO was proposed by Dorigo [3]. When we refer to the colonies of ants, we observe the ants communicate to each other just in an indirect form in their environment by the substance called pheromone. Paths with higher levels of pheromone will be likely to be chosen and consequently reinforced while the intensity of pheromone along the paths that are not chosen is reduced by evaporation. Additionally, evaporation causes the pheromone level of all trails to diminish gradually. Hence, trails that are not reinforced gradually lose pheromone and will in turn have a lower probability of being chosen by subsequent ants. Evaporation is accomplished by diminishing the pheromone level of each trail by a factor ρ . Typical values for this evaporation factor ρ lie in the range [0.8, 0.99][3]. This is an important mechanism to update the pheromone on the trails according to $\tau_{ij} \leftarrow (1 - \rho) * \tau_{ij} + \frac{Q}{TWCV}$, whose parameters are explained in more details in the next section. This form of indirect communication is known and gives the colony of ants the capacity for finding the shortest path [8]. There are some works related to the algorithms of clusters based on ACO. Yuqing et al. Proposes algorithm of K-means clusters based on density and on the Colony of Ants [13]. This algorithm is a new K-means algorithm based on the density and theory of ants, which solved the problem of the local minimum by the random ants, besides manipulating the initial parameters of K-means. Handl et al. proposes clustering based on ants [5].

3.3 Ant K-Means

The choice of this algorithm is because it produced satisfactory results regarding the clustering problem. In this method, It is necessary to provide the number of clusterings like in the conventional K-means algorithm for AK algorithm. Suppose $E = O_1, O_2, \dots, O_n$ the set of n data or objects, where O represents the objects collected from the database, in that each object has k attributes, where $k > 0$. Below some important parameters such as: α : The relative importance of the trail: $\alpha \geq 0$; β : The relative importance of the visibility: $\beta \geq 0$; ρ : The pheromone decay parameter: $0 < \rho < 1$; Q : A constant; n : Number of objects; m : Number of ants; nc : Number of clusters; T : is the set includes used objects. The maximal number recorded by T array will be n , i.e., $T = O_a, O_b, \dots, O_t$ where a, b, \dots, t are the points that ant has been. T_k : The set T is performed by ant k . $O_{center}(T)$: The object which is the center of all objects in T , i.e., $O_{center}(T) = \frac{1}{nT} \sum_{i=1}^{nT} O_i$, where nT is the number of objects in T . $TWCV$: total within cluster variance, i.e., $\sum_{k=1}^{nc} \sum_{i=1}^n (O_i - O_{center}(T_k))^2$.

The Algorithm 1 shows the procedure Ant K-means in details above.

Algorithm 1. The procedure of Ant K-means [7]

```

1 Procedure Perturbation: Each Ant starts at random object and chooses the
  centroid randomly of cluster to move for all Ant k. Calculating  $O_{center(T_k)}$ 
  where  $k = 1, 2, \dots, nc$  and TWCV.
2 Procedure Ant K-means: Input the number of clusters and the
  corresponding centroids, and set the parameter  $\alpha, \beta, \rho$ , number of iterations
  and ants. Lay equal pheromone on each path.
3 while the number of iterations is not reached do
4   while TWCV is not changed do
5     Updating pheromone by  $\tau_{ij} \leftarrow (1 - \rho) * \tau_{ij} + \frac{Q}{TWCV}$ .
6     Each Ant k chooses the centroid to move with P, i.e.,
7     
$$P = \frac{\tau_{kc}^{\alpha} * \eta_{kc}^{\beta}}{\sum_{i=1}^{nc} (\tau_{ki}^{\alpha} * \eta_{ki}^{\beta})}$$

8     Calculate  $O_{center(T_k)}$  where  $k = 1, 2, \dots, nc$ 
9     Calculate TWCV (Total Within Cluster Variance).
10    if TWCV is smaller than the smallest TWCV then
11      replace it.
12    else
13      Perturbation

```

3.4 SOM-Based Two-Stage Methods

A proposed method of clustering based on two stages is useful to improve the main disadvantages of a partitive method of clustering; for example, K-means due to its sensitivity to the initial prototypes and the difficulty in determining a proper number of k clusters. Generally, a SOM-based two-stage method has two possible forms of working. In the first one SOM is initially used to determine the number of groups and the center of the initial groups for the Ant K-means. The initial center of a group can be obtained from the weight vector corresponding to the center of the groups on the topology of SOM net. In the second form, the initial maps of SOM net present a large set of scale data on its topology and generates the topological coordinates of the prototypes for future clusters in the second stage. The method used in the second phase is the Ant K-means procedure. The main advantage of a SOM-based two-stage method is the reduction of the computational time by the hierarchical clustering method or partitive for the large and complex sets of data [12].

To show this characteristic, it was necessary to train a SOM net by using the algorithm of sequential training for the data set¹. The maps were trained in two phases: a rough training with width of initial neighborhood $\sigma_1(0)$ and big learning rate and another phase called fine-tuning with width of initial neighborhood

¹ Training of SOM net freely available in the package Matlab SOM Toolbox which was used in the implementation of the proposed method. For further information, see URL <http://www.cis.hut.fi/projects/somtoolbox/>

$\sigma_2(0)$ and small learning rate, which the width of the neighborhood decreases linearly to 1.

4 SOMAK

The method proposed in this paper, SOMAK, can be seen in Fig. 1. SOMAK uses SOM net as a classifier of characteristics about the entry data instead of clustering the data directly. First, a large set of prototypes is formed by using SOM. The prototypes can be interpreted as “proto-clusters”, which are in the next step combined to form the true clusters. Each data vector of the original data set belongs to the same cluster like its closest prototype. In the present study, the number of clusters and the centroid of each cluster are generated from SOM net. In order to validate the solution of clustering analysis, the framework Monte Carlo [9] was used in this paper. SOMAK uses SOM to determine the initial points and then uses the Ant K-means procedure to find out the final solution, i.e., AK to determine the number of clusters. The benefit of this approach is the reduction of the computational cost. The second advantage is the reduction of the clusters size. The reduction of the noise is another benefit. The prototypes are the local mean of the data and so, less sensitive to the random variations than the original data.

For this reason, it is convenient to cluster a set of prototypes, instead of the data directly [12]. Consider N samples of the data using Ant K-means algorithm which is described in section 3.3. This involves to make attempts of clustering with different values for the number of prototypes which were obtained by SOM net. The computational time is proportional to the $\sum_{k=2}^{C_{max}} Nk$, where C_{max} is the pre-established maximum number of clusters and k represents the number of initial clusters. When a set of prototypes is used in an intermediary step (Fig. 1 - 1st level of abstraction), the total time is proportional to $NM + \sum_{k=1}^{Mk}$, where M is the number of prototypes obtained. With $C_{max} = \sqrt{N}$ and $M = 5\sqrt{N}$, the reduction of the computational time is based on $\frac{\sqrt{N}}{15}$ or about six-fold for

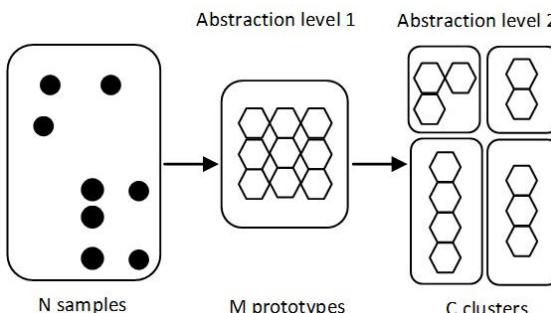


Fig. 1. The first level of abstraction is obtained through the creation of a set of prototype vectors by using SOM. Algorithm SOMAK creates the second level of abstraction carrying out the cluster of M prototypes [12].

$N = 10000$ [12]. In our case, we used ten-folds and $N = 1000$ for the carrying out of the experiments. Evidently this is a very rough estimate, since it is an estimate over the other; and many practical and experimental considerations are ignored.

5 Material and Methods

The carried out experiments were: synthetic data, real data, the method Monte Carlo, to check the efficiency of the four clustering methods. To carry out the experiments it was used a machine Intel(R) Core (TM) 2 Quad, processor 2.40GHz, memory RAM 3.00GB, operational system Microsoft Windows XP Professional version 2002 Service Pack 3.

5.1 Data Sets

In this paper, five data sets were used: *Lines*, *Banana*, *Highleyman* being these classified as synthetic data and *Contraceptive Method Choice* and *Glass* as real data.

The ***Lines*** basis consists of 1000 data points clustered in 10 segments. The other two bases of synthetic data² are arranged in the following way: A represents a data set of two classes in two dimensions; and N represents the number of samples of the vector generated with the number of samples by class. $N = [500, 500]$ having a total of 1000 data points. The ***Banana*** data basis shows too that the data points are distributed in a normal distribution in the form of a banana with standard deviation $S=1$ in all directions. Now the ***Highleyman*** third set of data, besides the arrangements mentioned before, is divided into two classes: the ***1st class*** contains 500 data points for each one of the Gaussians with mean 1 and 0 and variances 0 and 0.25, the ***2nd class*** contains 500 data points for each one of the two Gaussians with mean of 0.01 and 0 and variances 0 and 4. The real data used the repository UCI [1]. The ***Contraceptive Method Choice or CMC*** represents the problem of predicting the choice of a woman's current contraceptive method based on her economic and socio-demographic characteristics. The number of instances is 1473, divided into three classes. The number of attributes is 10, including the class. The second real basis is the ***Glass***, this database has as an objective to determine if some glass belongs to a kind "float" or not. The study of classification of this kind of glass was motivated by a criminological investigation in which several tests were made about the glass. The number of instances is 214, divided into six classes. The number of attributes is 11, including the class. All the synthetic data and the partitions of the real data were obtained through random numbers (Monte Carlo). After that, it was used the stratified cross validations with ten-folds on the databases providing the training and test sets for all the clustering methods.

² Data sets, freely available in the package Matlab PRTools: Toolbox for Pattern Recognition was used. For further information, see URL <http://prtools.org/academic.html>

So, it is reasonable to accept the reliability of the generator of random numbers. Finally, 30 executions about the project were carried out.

5.2 Parameters Setup

The parameters considered in this paper are those which affect direct or indirectly the clustering techniques, which were already described in the previously sections to solve the clustering problem. According to [3], there are several combinations to determine the parameters as applied to ant colony system. Normally, the parameters are $\alpha = 0, 0.5, 1, 2, 5$, $\beta = 0, 1, 2, 5$, $\rho = 0.3, 0.5, 0.7, 0.99, 0.999$ and $Q = 1, 100, 10000$. There are 300 combinations of parameters; the results showed in [2] that $\alpha = 0.5$, $\beta = 1$, $\rho = 0.9$ and $Q = 1$ in this method has the smallest variance, where $m = 2$ obtained the best results compared with $m = 4$ suggested by Marco Dorigo [3]. Table 1 shows the parameters of the clustering techniques.

Table 1. Main Parameters of Clustering Techniques

Clustering Techniques	Parameters
SOM	Attributes number = pattern quantity input, Lines size grid = 19, Columns = 17, Initial radius = 10, Final = 2, $\sigma_1(0) = 10$, $\sigma_2(0) = 2$, Initial learning rates were 0.5 and 0.05 respectively, Final = 0.99, Neighborhood function = Gaussian, Neighborhood format = hexa, Train type = epochs, Training size for rough phase = 3, fine-tuning phase = 10.
K-means	$k = \text{Number (nº)} \text{ initial clusters}$, Initialize centers = k .
SOMK	$k = \text{SOM prototypes nº}$, Initialize centers = SOM centroids $nº$.
SOMAK	$\alpha = 0.5, \beta = 1, \rho = 0.9, Q = 1, n = 500$, $m = 2, nc = \text{SOMprototypesnumber}$.

6 Experimental Results and Discussion

Then, to find the number of “proto-clusters” which obtains as a result 110 through SOM net; AK is used to cluster 500 data samples under the test set. Table 2 shows a comparison between SOMAK and SOMK to obtain a smaller number of clusters. The number of clusters and its centroids are obtained by SOM net and then uses AK to find the definite solutions. SOMAK has the best efficiency in comparison with SOMK, which is also the method composed of the two stages.

It is important to mention in Table 2 that the fact of the SOMAK method increases the number of clusters (compared to SOMK) does not mean to say that is bad, perhaps this increase may be necessary to have an improvement of entropy. SOMAK is applied as a technique of clustering for the case study because it obtained a smaller value of clusters. It will be presented the measure of Entropy, which showed a smaller value for SOMAK when compared to SOMK, the parameters Min, Max, Mea and Std represent respectively Minimum, Maximum, Mean and Standard Deviation in Table 3. The degree to which each cluster

Table 2. Results of the size of clusters obtained by the test set

Data sets	Initial	Cluster	SOMK	SOMAK
Lines(I)	10	6	3	
Banana(II)	2	7	4	
Highleyman(III)	2	3	4	
CMC(IV)	3	9	4	
Glass(V)	6	5	3	

Table 3. Results of the methods with 30 executions each to obtain the Entropy and the Computational Time (seconds)

Data sets	Methods	Results Entropy			Computational Time				
		Min	Max	Mea	Std	Min	Max	Mea	Std
I	SOM	0.043	0.154	0.103	0.028	1.764	1.811	1.791	0.012
	Kmeans	0.003	0.026	0.006	0.005	0.332	0.340	0.336	0.002
	SOMK	0.325	0.398	0.366	0.018	2.346	2.397	2.376	0.014
	SOMAK	0.229	0.341	0.273	0.026	1.931	1.999	1.970	0.016
II	SOM	0.036	0.122	0.074	0.021	1.764	1.847	1.818	0.014
	Kmeans	0.426	0.477	0.460	0.012	0.263	0.291	0.269	0.005
	SOMK	0.375	0.457	0.415	0.019	2.396	2.488	2.456	0.016
	SOMAK	0.267	0.421	0.340	0.041	1.939	2.035	1.995	0.019
III	SOM	0.212	0.326	0.268	0.029	1.795	1.837	1.819	0.010
	Kmeans	0.453	0.514	0.482	0.013	0.269	0.302	0.277	0.005
	SOMK	0.410	0.491	0.450	0.018	2.436	2.485	2.463	0.014
	SOMAK	0.250	0.448	0.365	0.057	1.955	2.019	1.990	0.015
IV	SOM	0.478	0.509	0.493	0.007	4.494	4.552	4.531	0.016
	Kmeans	0.397	0.450	0.413	0.012	0.292	0.308	0.300	0.005
	SOMK	0.407	0.467	0.442	0.013	5.267	5.333	5.304	0.019
	SOMAK	0.175	0.314	0.245	0.035	4.748	4.827	4.793	0.017
V	SOM	0.306	0.397	0.365	0.020	3.427	3.577	3.554	0.030
	Kmeans	0.286	0.368	0.331	0.023	0.333	0.351	0.341	0.004
	SOMK	0.295	0.370	0.337	0.020	3.945	4.107	4.080	0.032
	SOMAK	0.238	0.372	0.317	0.036	3.478	3.630	3.602	0.030

consists of objects of a single class. For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij} , the probability that a member of cluster i belongs to class j as $p_{ij} = \frac{m_{ij}}{m_i}$, where m_i is the number of objects in cluster i and m_{ij} is the number of objects of class j in cluster i . Using this class distribution, the entropy of each cluster i is calculated using the standard formula [14], $e_i = -\sum_{j=1}^L p_{ij} \log_2 p_{ij}$, where L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e., $e = \sum_{i=1}^K \frac{m_i}{m} e_i$, where K is the number of clusters and m is the total number of data points.

The Std parameter reported in Table 3 presented a smaller value for most of the methods of clusterings except for SOMAK. So, Table 3 showed also a

great variability pointed out in the standard deviation parameter, resulting in a disadvantage for the proposed SOMAK method. In the majority of the experiments, the SOMAK method showed a smaller entropy in the parameters of Min, Max and Mea, when compared with the methods of clustering SOM, K-means and SOMK for IV and V data sets. For the I, II and III data sets the SOMAK method presented a smaller entropy when compared with SOMK. Table 3 also shows computation time for all the techniques of clusterings used in the experiments. K-means has always been the quickest one computationally, because it is a simple algorithm or of only one stage. However, this same algorithm presented a high entropy seen in Table 3, when compared with the SOM, SOMK and SOMAK methods. SOMAK had a longer time than K-means and obtained more satisfactory results when compared with SOMK as well in reference to the entropy as to the computational time.

It was concluded that the experimental results are statistically independent according to the application of Test t (hypothesis test). It was applied as well for the entropy as for the computational time respectively seen in Table 3 and with 5% of significance degree it showed that SOMAK is better than SOMK.

7 Conclusion and Future Work

The aim of this paper was to propose a method of clustering, SOMAK, composed of two stages by combining SOM and Ant k-means. The SOMAK method is capable of reducing the size of clusters, by finding a good performance when compared with other techniques of clustering (SOM, K-means and SOMK) and also capable of reducing the computational time of the experiments.

The algorithms of clusters described before were tested as well for the data directly as for the data trained by SOM net. It was used a SOM net as an intermediary step besides carrying out a comparison of the results obtained directly from the data. The results for the data generated by the Monte Carlo method showed that SOMAK is better than SOMK, because there was a reduction of the size of the clusters for the test set (Table 2), for it to have formed a better performance when compared with SOMK seen (Table 3) and finally, Table 3 also shows that SOMAK reduced the computational time in comparison with SOMK to solve the problem of data clusterings. So, the proposed method is a robust method of clustering. It can be applied to a lot of different kinds of clustering problems or combined with some other techniques of data mining to obtain more promising results.

For future works, the idea is readjusting the SOMAK algorithm with the purpose of reducing its computational time when compared with the methods described in this paper. The first method that will be observed is the ABSOM [11]. This one has better performance than SOM and also works very well in the analysis of clustering in two stages when it is used as a technique of pre-processing. So, this method is composed of two stages for the analysis of data, where it has demonstrated to be useful and effective.

Acknowledgments

This work is supported by CNPq and FACEPE (Brazilian research agencies).

References

1. Aha, D.: UCI machine learning repository, <http://archive.ics.uci.edu/ml/> (access January 16, 2009)
2. Berkhin, P.: Survey of Clustering Data Mining Techniques, Accrue Software, <http://www.accrue.com/> (access January 7, 2009)
3. Dorigo, M., Stützle, T.: The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, Technical Report IRIDIA (2000)
4. Everitt, B.S., Landau, S., Leese, M.: Cluster Analysis. Edward Arnold, London (2001)
5. Handl, J., Knowles, J., Dorigo, M.: Ant-Based Clustering: A Comparative Study of its relative performance with respect to k-means, average link and 1D-SOM, IRIDIA-Technical Report Series (2003)
6. Kohonen, T.: The self-organizing map. Neurocomputing 21, 1–6 (1998)
7. Kuo, R.J., Wang, H.S., Hu, T.-L., Chou, S.H.: Application of Ant K-Means on Clustering Analysis. Computers and Mathematics with Applications 50(10-12), 1709–1724 (2005)
8. Martens, D., De Backer, M., Haesen, R.: Classification with Ant Colony Optimization. IEEE Transactions on Evolutionary Computation 11(5), 651–665 (2007)
9. Milligan, G.W.: An Algorithm for generating Artificial Test Clusters. Psychometrika 50(1), 123–127 (1985)
10. Mitchell, T.: Machine Learning, 352 p. McGraw-Hill, New York (1997)
11. Sheng-Chai, C., Chih-Chieh, Y.: A Two-stage Clustering Method Combining Ant Colony SOM and K-means. Journal of Information Science and Engineering 24, 1445–1460 (2008)
12. Vesanto, J., Alhoniemi, E.: Clustering of the Self-Organizing Map. IEEE Transactions on Neural Networks 11(3), 586–600 (2000)
13. Yuqing, P., Xiangdan, H., Shang, L.: The k-means clustering algorithm based on density and ant colony. IEEE Intelligent Neural Networks and Signal Processing 1, 14–17 (2003)
14. Tan, P., Steinbach, M., Kumar, V.: Introduction to Data Mining, 769 p. Pearson, London (2006)

Image Theft Detection with Self-Organising Maps

Philip Prentis, Mats Sjöberg, Markus Koskela, and Jorma Laaksonen

Czech Technical University in Prague,

Helsinki University of Technology

prentphi@fjfi.cvut.cz,

{mats.sjoberg,markus.koskela,jorma.laaksonen}@tkk.fi

<http://www.cvut.cz>, <http://www.tkk.fi>

Abstract. In this paper an application of the TS-SOM variant of the self-organising map algorithm on the problem of copyright theft detection for bitmap images is shown. The algorithm facilitates the location of originals of copied, damaged or modified images within a database of hundreds of thousands of stock images. The method is shown to outperform binary decision tree indexing with invariant frame detection.

Keywords: Image theft detection, image retrieval, self-organising maps, TS-SOM, PicSOM.

1 Introduction

Companies that publish large on-line databases of stock images are often faced with the problem of copyright infringement due to image theft. This occurs when people copy images and reuse them for commercial purposes without paying. Locating such images on the web and proving that they are stolen from the company's database may be difficult if the database is very large or if the images have been modified.

One of the few published attempts at solving this problem is presented in [12], where Horacek et al describe a method of copyright theft detection that uses stochastic decision trees, which is used as a comparison method in this paper.

Other attempts include *content-based image detection* [3] and alternative means of dealing with image theft such as watermarking (e.g. [4]).

In the past, self-organising maps [5] have been used on a number of image processing and retrieval tasks such as content-based image retrieval [8,9,10,11,12], automatic image annotation [14], colour-based image browsing [15] and others. The use of tree-structured variants, which allow fast logarithmic search [6,7], therefore presents itself for copyright theft detection.

2 Methodology

2.1 Self-Organising Maps

Simply put, a self-organising map (SOM) is a structure resulting from an iterative algorithm that reduces a high-dimensional input-space (set of vectors

from R^N) to a two-dimensional ordered grid of codebook vectors (neurons) $\{n_{ij} | i = 1..N_1, j = 1..N_2; \text{typically } N_1 = N_2\}$ that quantify it. Each input vector is quantised by the codebook vector with the smallest Euclidean distance.

In a successfully adapted map, adjacent grid nodes will quantise similar data, i.e. data points that have a small Euclidean distance. The principle may be seen in Fig. 1, which shows a typical SOM in a 2D input space.

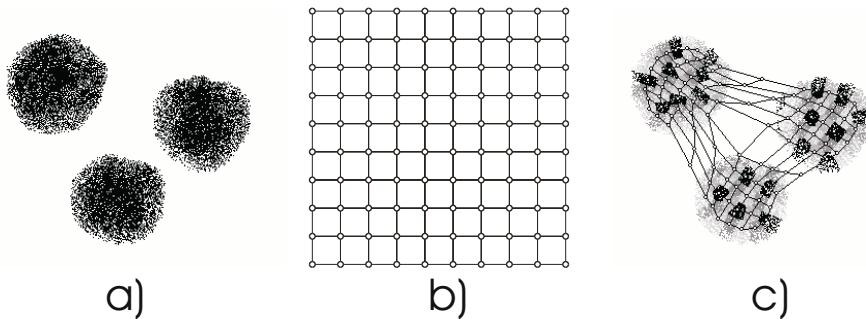


Fig. 1. a) 2-dimensional input vectors in 3 clusters. b) A SOM, topologically ordered in a grid of 10×10 neurons c) The SOM adapts itself to match the input space, each neuron's codebook vector quantifying a set of inputs, denoted by differing shades of grey.

Algorithm. The self-organisation process is achieved as follows:

1. Initialise the codebook vectors $n_{ab}(0)$ at random (usually by setting them to randomly chosen input vectors).
2. Select a random input $i(t)$ and find the best matching neuron (BMN) $n_{best}(t)$ (i.e. the neuron with the closest codebook vector). Every input sample has the same probability of being selected.
3. Move the BMN and its topological neighbours within a certain neighbourhood distance towards the selected input vector. Units located topologically further from BMN are moved less.

$$n_{ab}(t+1) = n_{ab}(t) + \eta(t) \cdot \phi(a, b, t) \cdot [i(t) - n_{ab}(t)], \quad (1)$$

where

$$\eta(t) : N_0 \rightarrow [0; 1] \quad \text{monotonously decreasing}, \quad (2)$$

$$\phi(a, b, t) : N_0 \times N_0 \times N_0 \rightarrow [0; 1],$$

ϕ decreases monotonously with the topological distance of n_{ab} from n_{best} and with t . The topological distance is usually calculated as the length of the shortest path from one neuron to the other in the graph (grid) that represents the network's topology.

4. Proceed to iteration $t+1$. Repeat 2 and 3 iteratively, reducing the proportion of the distance moved η and the neighbourhood distance ϕ each iteration, until they reach a certain predetermined threshold.

It should be noted that problems may occur if the map does not adapt properly and the topology of the map is not be preserved. In such cases similar data may be classified by topologically distant neurons causing classification errors. For more on self-organising maps, see [5].

2.2 Tree-Structured Self-Organising Maps

With larger maps, search for the BMN slows quadratically with the growing width/height of the topology. The complexity of the search is $O(N)$, where N is the number of neurons in the map (each neuron must be compared to determine the BMN). One solution to this problem is to use tree-structured self-organising maps (TS-SOM), [67]. A TS-SOM is a hierarchical structure of SOMs of exponentially increasing size. Each level of the TS-SOM adapts separately, but in the lower levels, the search for the best-matching neuron is limited to those hierarchically connected to the BMN of the previous layer and their neighbours. See Fig. 2.

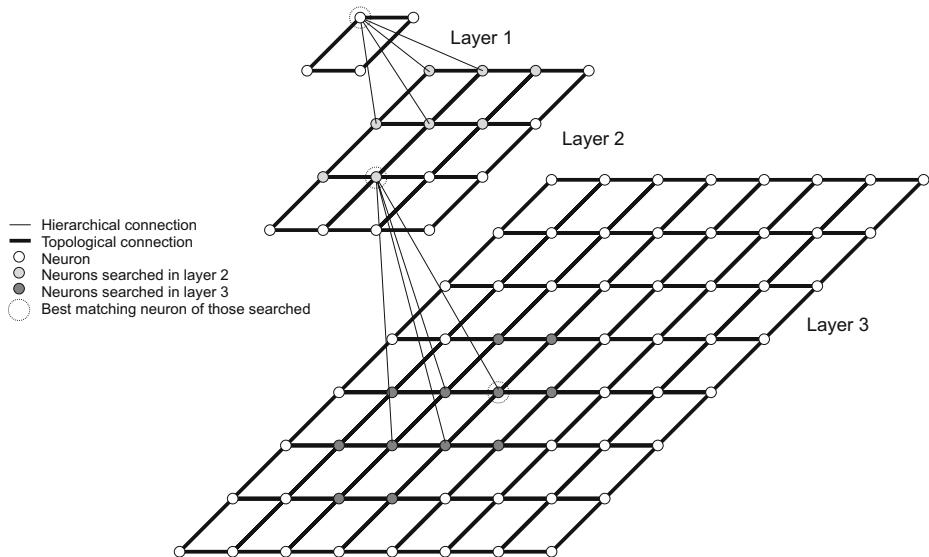


Fig. 2. A 3-layer TS-SOM with 4 neurons at top layer and 64 at the bottom

Algorithm

1. Iterate the SOM algorithm on the top layer until the threshold has been met.
2. Iterate the SOM algorithm on the next layer until the threshold has been met, but limit the search for the BMN to the neurons located under the winning neuron of the previous layer.
3. Repeat 2 until all layers have been updated.

The advantages of such a structure are obvious. Instead of performing a full-search for the BMN at the lower layers, we restrict ourselves to a constant number of neurons per a given layer, thus greatly increasing the adaptation speed. The complexity of the algorithm is $O(\log N)$, where N is the number of neurons on the bottom layer [7]. Also, due to the hierarchical structuring, all the SOMs will be orientated similarly in input space.

2.3 PicSOM

PicSOM [10] works by taking a number of TS-SOMs, each one mapping a different feature calculated on the given image set. A query image is taken and its BMN is located in each map. (In classification tasks, multiple queries are used, so multiple BMN are located; however, for copyright theft detection, only a single query is used.) For the lowest-level layers of the TS-SOM, matrices of values are calculated, one value for every neuron. The BMN is awarded a value of 1 while the others are given a value of 0. Then a convolution filter with a triangular kernel (not to be confused with the kernel used in the SOM algorithm) is passed over it, increasing the value of those neurons within the vicinity of the BMN. Due to the topology preserving property of the SOM these nearby units can be expected to be similar to the BMN. Finally, the result is normalised. See Fig. 3.

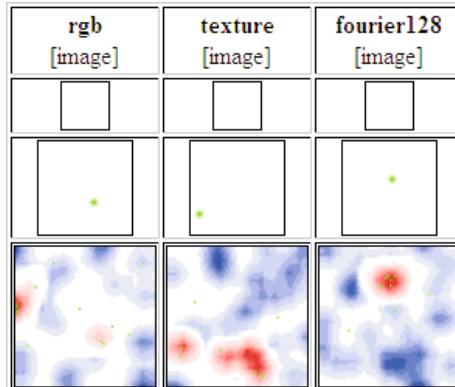


Fig. 3. PicSOM: Three 3-layer TS-SOM with 16 neurons at the top layers and 4096 at the bottom. Value maps are calculated for the bottom layer, low values being coloured blue and high values red. BMN of queries are marked in green.

PicSOM retrieves a desired number of closest matching images from the database according to the sum of their value scores on each map. Therefore, images that score highly on multiple maps (and are therefore similar to the query in multiple features) will be chosen with higher priority than those on fewer maps with lower scores.

3 Experiments

3.1 Data

Image Database. The experiments were run on feature vectors calculated from a data set of 322283 stock images. Each image consisted of a jpeg-encoded true-colour bitmap, scaled to fit in a 640×480 pixel window.

Stolen Images. Stolen images were simulated by taking the first 1000 images in the database and modifying them using 7 different combinations of randomly varying degrees of the following distortions:

- Radiometric distortion (contrast, brightness)
- Scaling
- Cropping
- Frame added
- Logo added

The 7 combinations were as follows:

1. RndLogo1 – random logo
2. RndScale1 – random scaling
3. RndRadiom1 – random radiometric distortion
4. RndCrop1 – random cropping
5. RndLogSca1 – random logo + scaling
6. RndFraLogSca1 – random frame + logo + scaling
7. RndLogCroRad1 – random logo + cropping + radiometric distortion

The 7000 resulting *stolen images* were then used as queries for testing the method's precision.

3.2 Feature Selection

Two experiments were performed. In the first the standard selection of features used by PicSOM [13] was used.

1. rgb = Average rgb colour: average values of Red, Green and Blue channels, calculated in five zones.
2. qgreyrgb = Average grey-scale intensity: as above, but calculated for average grey-scale intensity of zones rather than separate rgb channels.
3. texture = Texture neighbourhood: texture feature based on brightness of neighbouring pixels, calculated in five zones.
4. z1texture = Single zone texture neighbourhood: as above, but counting the entire image as a single zone.
5. colm = Colour moments: values of three first central moments of colour distribution.
6. edgehist = Edge histogram: histogram of four Sobel edge directions, calculated in five zones.

7. edgecoocc = Edge co-occurrence matrix: co-occurrence matrix of four Sobel edge directions, calculated in five zones.
8. colourlayout = Colour layout: DCT coefficients of average colour in 20×20 grid.
9. scalablecolour = Scalable colour: Haar transform of quantised HSV colour histogram.
10. dominantcolour = Dominant colour: CIE Lab coordinates of three dominant colour clusters.
11. edgehistogram = Edge histogram: histogram of five edge types in 4×4 subimages.

Of these, `rgb`, `qgreyrgb`, `texture`, `colm`, `edgehist` and `edgecoocc` are calculated for five separate, equally-sized zones of the given image – a circular zone at the centre, taking up one fifth of the area, plus the top, bottom, left and right portions of the remainder. See Fig. 4

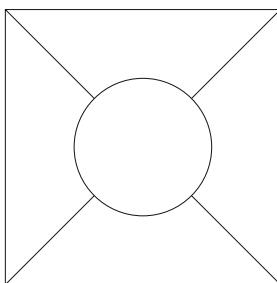


Fig. 4. For the calculation of the `rgb`, `texture`, `colm`, `edgehist` and `edgecoocc` features in PicSOM, images are divided up into five separate, equally-sized zones

In the second experiment, only the centre zones of these features were calculated in an attempt to improve the results for images with added frames (with the exception of `qgreyrgb`, which would have been reduced to a scalar value).

3.3 Precision Measure

Precision was calculated for each of the 7 combinations of image distortions separately as the percentage of originals located by the method. An image was considered to be located by the method if it appeared within the first 20 images returned by PicSOM. It is assumed that such a small number of images could be easily visualised at once as thumbnails and identified by a human operator or safely verified using a computationally heavier method such as phase-correlation, as in [1].

3.4 TS-SOM Parameters

The TS-SOMs used for mapping the feature spaces describing the data were made of 4 levels of 4×4 , 16×16 , 64×64 and 256×256 neurons, respectively. Each neuron on the top three layers was hierarchically connected with the 4×4 neurons located directly beneath it on the next layer with the search for the BMN being conducted on an area of 10×10 neurons beneath it – i.e. a frame 3 neurons wide surrounding the 16 hierarchically connected ones (the area was 7×10 for edge neurons and 7×7 for corner ones, as the map topology did not loop).

Each map was trained by performing 100000 iterations on each level in succession. To prevent overlearning, the training parameters were not fine-tuned to match the data, but simply taken from previous unpublished optimisation performed on other data.

3.5 Training Time

Feature Vectors. The average calculation time of the individual features ranged from 0.17s (e.g. 11 – Edge histogram) to 0.67s (e.g. 10 – Dominant colour) per image, depending on the complexity of the features used. The average total calculation time for all features per a single image was 3.51s. Therefore, with parallelization, feature vectors for the entire database of 323 thousand images could be calculated in approximately 60 hours. If calculated in series, it would take about 2 weeks.

TS-SOM Training. TS-SOM training time varied from a couple of hours to several days, depending on the dimensionality of the feature vectors.

Search Time. Due to the fast logarithmic search for the BMN used by the TS-SOM, the search time for a single *stolen image* is insignificant when compared with the feature vector calculation time (3.51s), which precedes it.

3.6 Comparison

The results were compared with those of the binary decision tree (BDT) method used in [1], which were calculated on the same 7000 “stolen images” with originals in an unspecified subset of 100000 of the 322283 images in the database. Their precision scores are equal to the percentage of located images. An image was considered located if it was one of the top twenty returned by the BDT indexing system and had been successfully verified with phase correlation.

As there were only *features* calculated from the original images available (i.e. the actual images were not available), it was not possible to incorporate verification process into PicSOM. However, cases of incorrect phase correlation verification and subsequent loss of precision were sufficiently rare as to render them insignificant.

3.7 Results

Table II shows the average precision scores calculated on the 1000 images of a given combination of transformations for both experiments (PicSOM with standard features and PicSOM with centre-zone only features) as well as the comparison scores of the Binary Decision Tree Indexing system published in [4]. As can be seen, PicSOM achieved better results than BDT for almost all combinations of image distortion, with the exception of framed images where BDT scored marginally higher. The most significant difference was for cropped images, where PicSOM greatly outperformed BDT.

Furthermore, limiting the calculation of the features to the centre zones of the images greatly improved the results for framed images (outperforming BDT by 17.6%), while achieving similar albeit slightly reduced precision scores for the other distortion types.

Table 1. PicSOM – PicSOM retrieval with standard feature set; PicSOM-C – PicSOM retrieval using centre-zones of 5-zone features; BDT – Binary Decision Tree with invariant frame detection (comparison). Highest average precision for given transformation combination marked in bold, lowest in italics.

Average Precision									
Stolen image set		Method		Transformations used on stolen images					
No.	Name	PicSOM	PicSOM-C	BDT	Radiom.	Scale	Crop	Frame	Logo
1	RndLogo1	0.998	0.999	<i>0.982</i>					Y
2	RndScale1	0.998	0.998	<i>0.946</i>		Y			
3	RndRadiom1	0.792	0.770	<i>0.712</i>	Y				
4	RndCrop1	0.762	0.721	<i>0.450</i>			Y		
5	RndLogSca1	0.997	0.993	<i>0.934</i>		Y			Y
6	RndFraLogSca1	<i>0.300</i>	0.534	0.358		Y		Y	Y
7	RndLogCroRad1	0.365	0.327	<i>0.182</i>	Y		Y		Y

4 Conclusions and Discussion

It should be noted that despite the results being skewed in favour of the BDT comparison method (due to the different database sizes), PicSOM still managed to outperform it in all cases without recourse to any specialised feature selection (the centre-zone features were simply a subset of the data provided by the standard feature set). No high-level features (i.e. object detection, etc.) or more sophisticated calculations such as invariant frame detection were used.

As both methods use logarithmic search (binary tree vs. TS-SOM) and the constant time for the preprocessing of a single query image (feature calculation vs. invariant frame detection) is around 3s in both cases, neither method may be concluded to be objectively faster.

It is interesting to note how the reduction of the feature set to centre zones dramatically improves the precision for framed images without significantly damaging the other results. Otherwise, framed images were the only case where – thanks

to the invariant frame detection – the BDT indexing method would have outperformed PicSOM.

In summary, in this paper we have described the problem of image copyright theft detection and proposed a solution using tree-structured self-organising maps (TS-SOM). We have shown that TS-SOM using low-level features can outperform binary decision trees using invariant frame detection. Further research may include experimentation with more specialised features on larger databases and/or real world data.

Acknowledgments

We would like to thank the Institute of Information Theory and Automation of the Academy of Sciences of the Czech Republic for kindly providing access to features calculated on their database of stock images.

Mats Sjöberg has been supported by a grant from the Nokia Foundation.

References

1. Horáček, O., Bican, J., Kamenický, J., Flusser, J.: Image Retrieval for Image Theft Detection. In: The 5th International Conference on Computer Recognition Systems, Warsaw (2007)
2. Horáček, O., Bican, J., Kamenický, J., Flusser, J.: Image Retrieval for Image Theft Detection. In: Computer Recognition Systems 2. Advances in Soft Computing, vol. 45, pp. 44–51. Springer, Heidelberg (2008)
3. Kim, C.: Content-based image copy detection. *Signal Processing: Image Communication* 18(3), 169–184 (2003)
4. Craver, S.: Zero Knowledge Watermark Detection. In: Pfitzmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 101–116. Springer, Heidelberg (2000)
5. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Berlin (2001)
6. Koikkalainen, P., Oja, E.: Self-organizing hierarchical feature maps. In: The International Joint Conference on Neural Networks, San Diego, California, vol. II, pp. 279–284 (1990)
7. Koikkalainen, P.: Progress with the tree-structured self-organizing map. In: The 11th European Conference on Artificial Intelligence (1994)
8. Laaksonen, J., Koskela, M., Laakso, S., Oja, E.: Self-Organizing Maps as a Relevance Feedback Technique in Content Based Image Retrieval. *Pattern analysis & Applications* 4(2-3), 140–152 (2001)
9. Laaksonen, J., Koskela, M., Oja, E.: Application of Self-Organizing Maps in Content Based Image Retrieval. In: The 9th International Conference on Neural Networks, Edinburgh, (1999).
10. Laaksonen, J., Koskela, M., Oja, E.: PicSOM - Self-Organizing Image Retrieval With MPEG-7 Content Descriptors. *IEEE Transactions on Neural Networks* (2002)
11. Laaksonen, J., Koskela, M., Laakso, S., Oja, E.: PicSOM - content-based image retrieval with self-organizing maps. *Pattern Recognition Letters* 21 (2000)
12. Laaksonen, J., Koskela, M., Laakso, S., Oja, E.: The PicSOM Retrieval System: Description and Evaluations. In: Proceedings of CIR-2000, Brighton, UK (2000)

13. Laaksonen, J., Oja, J., Koskela, M., Brandt, S.: Analyzing Low-level Visual Features Using Content-Based Image Retrieval. In: The 7th International Conference on Neural Information Processing, Taejon, Korea (2000)
14. Viitaniemi, V., Laaksonen, J.: Keyword-detection approach to automatic image annotation. In: Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies, London, UK (2005)
15. Prentis, P.: GalsOM - Colour-Based Image Browsing and Retrieval with Tree-Structured Self-Organising Maps. In: The 6th International Workshop on Self-Organizing Maps, Bielefeld, Germany (2007)

Improved Kohonen Feature Map Associative Memory with Area Representation for Sequential Analog Patterns

Tomonori Shirotori and Yuko Osana

Tokyo University of Technology,
1404-1 Katakura Hachioji, Tokyo, Japan
osana@cs.teu.ac.jp

Abstract. In this paper, we propose an improved Kohonen feature map associative memory with area representation for sequential analog patterns. This model is based on the conventional Kohonen feature map associative memory with area representation for sequential analog patterns. The proposed model has enough robustness for noisy input and damaged neurons. Moreover, the learning speed of the proposed model is faster than that of the conventional model. We carried out a series of computer experiments and confirmed the effectiveness of the proposed model.

Keywords: Kohonen Feature Map (Self-Organizing Map), Associative Memory, Successive Learning, Sequential Analog Pattern.

1 Introduction

Recently, neural networks are drawing much attention as a method to realize flexible information processing. In the field of neural networks, although a lot of models have been proposed, their learning and recall processes are divided, and therefore they need all information to learn in advance [1]–[4].

However, in the real world, it is very difficult to get all information to learn in advance, so we need the model whose learning process and recall process are not divided. As such model, some models have been proposed [5]–[11]. However, their storage capacities are small because their learning algorithm is based on the Hebbian learning.

On the other hand, the Kohonen Feature Map (KFM) associative memory [2] has been proposed. Although the KFM associative memory is based on the local representation, it can learn new patterns successively [13], and its storage capacity is larger than that of models in refs. [5]–[11]. It can deal with auto and hetero associations and the associations for plural sequential patterns including common terms [14]. Moreover, as the model which has robustness for damaged neurons, the KFM associative memory with area representation [15] has been proposed. The area representation [16] is an intermediate representation of the

local representation and the distributed representation. In the area representation, one concept is expressed by the winner neurons and some neurons located adjacent to the winner neuron.

As the model which can deal with sequential patterns, the KFM associative memory for temporal sequences [14] and the KFM associative memory with area representation for sequential patterns [17] have been proposed. In these models, an association for sequential patterns which have common terms is realized by using recurrent difference vectors. However, they can deal with only binary (bipolar) patterns. As the model which can deal with sequential analog patterns, we have proposed the KFM associative memory with area representation for sequential analog pattern [18].

In this paper, we modify the winner neuron selection method and the connection weights update method of the conventional KFM associative memory with area representation for sequential analog patterns, and propose an improved KFM associative memory with area representation for sequential analog patterns. This model has enough robustness for noisy input and damaged neurons. Moreover, the learning speed of the proposed model is faster than that of the conventional model.

2 KFM Associative Memory with Area Representation for Sequential Analog Patterns

Here, we explain the conventional KFM associative memory with area representation for sequential analog patterns [18].

2.1 Structure

Figure 1 shows the structure of the conventional KFM associative memory with area representation for sequential analog patterns. As seen in Fig. 1, it has two layers; (1) Input/Output Layer and (2) Map Layer, and the Input/Output Layer is divided into two parts; (1) Input Part and (2) Output Part.

2.2 Learning Process

The learning process of the conventional model is based on the conventional learning algorithm for the KFM associative memory with area representation for sequential patterns.

Let $\mathbf{Y}^{(k,1)} \rightarrow \mathbf{Y}^{(k,2)} \rightarrow \dots \rightarrow \mathbf{Y}^{(k,t_k)}$ be the k th temporal sequence to be stored, where t_k shows the length of the k th sequence. Then, the learning vectors $\{\mathbf{X}^{(k,t)}\}_{k=1,\dots,K}$ are defined by

$$\mathbf{X}^{(k,t)} = \begin{pmatrix} \mathbf{Y}^{(k,t)} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{Y}^{(k,t+1)} \end{pmatrix} (t = 1, \dots, t_k - 1). \quad (1)$$

In the sequential learning algorithm for the conventional KFM associative memory with area representation for sequential analog patterns, the connection weights are learned as follows:

- (1) The initial values of weights are chosen randomly and the recurrent difference vector is set to $\mathbf{y}_i = \mathbf{0}$.
- (2) The recurrent difference vector of the neuron i in the Map Layer $\mathbf{y}_i(n, t)$ is calculate by

$$y_{ij}(n, t) = \begin{cases} (1 - \beta)y_{ij}(n, t - 1) + \beta(X_j^{(k,t)} - W_{ij}(n, t)) & (1 \leq j \leq N/2) \\ X_j^{(k,t)} - W_{ij}(n, t) & (N/2 < j \leq N) \end{cases} \quad (2)$$

where β ($0.5 < \beta < 1$) is the weighting factor determining the effect of the earlier difference vectors and the new input vector in the computation of $\mathbf{y}_i(n, t)$, and n is the number of learning iterations.

- (3) The winner neuron r whose recurrent difference vector $\|\mathbf{y}_i(n, t)\|$ is minimum is found.
- (4) The connection weights between the neurons in the Input/Output Layer and the neuron i in the Map Layer except those of fixed neurons are updated by

$$\mathbf{W}_i(n, t + 1) = \mathbf{W}_i(n, t) + H(d_i)\alpha(n)h_{ri}\mathbf{y}_i(n, t) \quad (3)$$

where $H(d_i)$ is given by

$$H(d_i) = \frac{1}{1 + \exp(-(d_i - D)/\varepsilon)} \quad (4)$$

In this equation, d_i is the Euclid distance between the neuron i and the nearest weights fixed neuron in the Map Layer, D is the constant for area size and ε is the steepness parameter of the function $H(d_i)$.

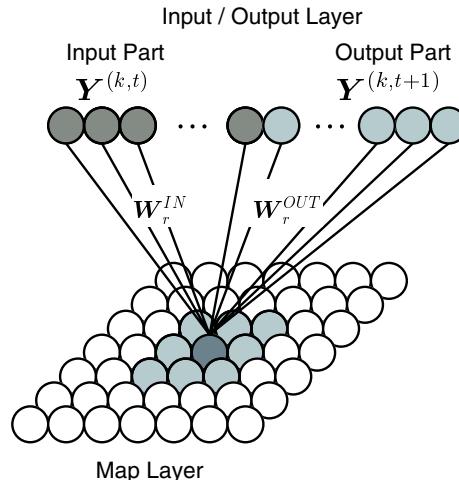


Fig. 1. Structure of KFM Associative Memory with Area Representation for Sequential Analog Patterns

- (5) (2)~(4) are iterated until $t = t_k - 1$. This accomplishes the learning of the k th temporal sequence on time. Then, for the next iteration, the recurrent difference vector is reseted as $\mathbf{y}_i = \mathbf{0}$. The obtained weights are inherited in the next iteration.

$$\mathbf{W}_i(n+1, 1) = \mathbf{W}_i(n, t_k) \quad (5)$$

- (6) (2)~(5) are iterated until $n = n_{max}$. Then, the weights of $t_k - 1$ winner neurons selected in the final iteration.
(7) (2)~(6) are repeated for all k .

Since the conventional KFM associative memory with area representation for sequential analog patterns is learned by using weights fixed and semi-fixed neurons, it can store a new temporal sequence without retraining previously learned temporal sequences.

2.3 Recall Process

When the pattern \mathbf{X} is given, the output of the neuron i in the Map Layer at the time t , $x_i^{map}(t)$ is given by

$$x_i^{map}(t) = \begin{cases} 1, & (\|\mathbf{y}_i(t)\| < \theta^{map}) \\ 0, & (\text{otherwise}) \end{cases} \quad (6)$$

$$y_{ij}(n, t) = \begin{cases} (1 - \beta)y_{ij}(n, t - 1) + \beta(X_j^{(k, t)} - W_{ij}(n, t)) & (1 \leq j \leq N/2) \\ 0 & (N/2 < j \leq N) \end{cases} \quad (7)$$

where θ^{map} is the threshold of the neuron in the Map Layer.

The output of the neuron j in the Input/Output Layer at the time t , $x_j^{in}(t)$ is given by

$$x_j^{in}(t) = \frac{1}{\sum_i x_i^{map}} \sum_{i:x_i=1} W_{ij}. \quad (8)$$

3 Improved KFM Associative Memory with Area Representation for Sequential Analog Patterns

Here, we explain the proposed improved KFM associative memory with area representation for sequential analog patterns. This model is based on the conventional KFM associative memory with area representation for sequential analog patterns [18] described in [2].

3.1 Structure

Figure 2 shows the structure of the proposed improved KFM associative memory with area representation for sequential analog patterns. As seen in Fig. 2, it has two layers; (1) Input/Output Layer and (2) Map Layer, and the Input/Output Layer is divided into two parts; (1) Input Part and (2) Output Part as similar as the conventional model. In the proposed model, as shown in Fig. 2, the Map Layer is treated as torus.

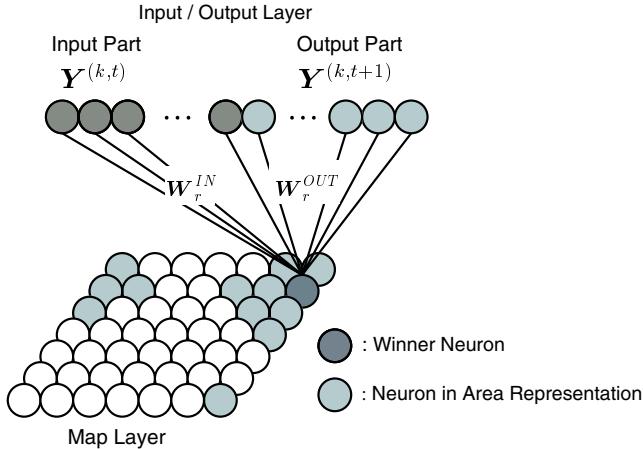


Fig. 2. Structure of Proposed Model

3.2 Learning Process

Let $\mathbf{Y}^{(k,1)} \rightarrow \mathbf{Y}^{(k,2)} \rightarrow \dots \rightarrow \mathbf{Y}^{(k,t_k)}$ be the k th temporal sequence to be stored, where t_k shows the length of the k th sequence.

In the sequential learning algorithm for the proposed improved KFM associative memory with area representation for sequential analog patterns, the connection weights are learned as follows:

- (1) The initial values of weights are chosen randomly and the recurrent difference vector is set to $\mathbf{y}_i = \mathbf{0}$.
- (2) The recurrent difference vector of the neuron i in the Map Layer $\mathbf{y}_i(n, t)$ is calculate by

$$y_{ij}(n, t) = \begin{cases} (1 - \beta)y_{ij}(n, t - 1) + \beta(X_j^{(k,t)} - W_{ij}(n, t)), & (j \leq M/2) \\ X_j^{(k,t)} - W_{ij}(n, t), & (\text{otherwise}) \end{cases} \quad (9)$$

where β ($0.5 < \beta < 1$) is the weighting factor determining the effect of the earlier difference vectors and the new input vector in the computation of $\mathbf{y}_i(n, t)$, and n is the number of learning iterations.

- (3) The winner neuron r is determined as follows:

$$r = \operatorname{argmin}_i (\|\mathbf{y}_i(n, t)\| (1 - s H^{\text{learn}}(d_{ii^*}))) \quad (10)$$

where s ($0 < s < 1$) is the coefficient. d_{ii^*} is the distance between the neuron i and the nearest weights fixed neuron i^* . In the proposed model, the map layer is treated as torus, so the distance between the neurons i and j , d_{ij} is given by

$$d_{ij} = \sqrt{(d_{ij}^x)^2 + (d_{ij}^y)^2} \quad (11)$$

$$d_{ij}^x = \begin{cases} |x_i - x_j|, & (|x_i - x_j| \leq x_{max}/2) \\ x_{max} - |x_i - x_j|, & (\text{otherwise}) \end{cases} \quad (12)$$

$$d_{ij}^y = \begin{cases} |y_i - y_j|, & (|y_i - y_j| \leq y_{max}/2) \\ y_{max} - |y_i - y_j|, & (\text{otherwise}) \end{cases} \quad (13)$$

where x_i and y_i are the coordinates of the neuron i in the Map Layer, x_j and y_j are the coordinates of the neuron j in the Map Layer, and x_{max} and y_{max} are width and height of the Map Layer.

In Eq.(10), $H^{learn}(d_{ii^*})$ is given by

$$H^{learn}(d_{ii^*}) = \frac{1}{1 + \exp\left(-\frac{d_{ii^*} - D^t}{\varepsilon^t}\right)} \quad (14)$$

where D^t is the constant which decides area size and ε^t is the steepness parameter.

- (4) The connection weight between the neurons in the Input/Output Layer and the neuron i in the Map Layer except those of fixed neurons are updated by

$$\mathbf{W}_i(n, t+1) = \mathbf{W}_i(n, t) + H(d_{ii^*})h_{ri}\mathbf{y}_i(n, t) \quad (15)$$

where h_{ri} is the neighbor function, and is given by

$$h_{ri} = \exp\left(\frac{-\|\mathbf{r} - \mathbf{i}\|^2}{2\sigma(n)^2}\right) \quad (16)$$

where $\sigma(n)$ is given by

$$\sigma(n) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{n/n_{max}} \quad (17)$$

where $\sigma(n)$ is set to σ_i at $n = 0$, and $\sigma(n)$ is set to σ_f at $n = T$ ($\sigma_i > \sigma_f$).

- (5) (2)~(4) are iterated until $t = t_k - 1$. This accomplishes the learning of the k th temporal sequence on time. Then, for the next iteration, the recurrent difference vector is reseted as $\mathbf{y}_i = \mathbf{0}$. The obtained weights are inherited in the next iteration.

$$\mathbf{W}_i(n+1, 1) = \mathbf{W}_i(n, t_k) \quad (18)$$

- (6) (2)~(5) are iterated until $n = n_{max}$. Then, the weights of $t_k - 1$ winner neurons selected in the final iteration.
(7) (2)~(6) are repeated for all k .

Since the proposed improved KFM associative memory with area representation for sequential analog patterns is learned by using weights fixed and semi-fixed neurons, it can store a new temporal sequence without retraining previously learned temporal sequences.

3.3 Recall Process

The recall process of the proposed model is same as the conventional KFM associative memory with area representation for sequential analog patterns.

4 Computer Experiment Results

Here, we show the computer experiment results to demonstrate the effectiveness of the proposed model.

4.1 Association Result

In this experiment, two sequential analog patterns shown in Fig. 3 were memorized successively. Figure 4 shows the association result of the proposed model. As shown in Fig. 1 when “fox” was given to the network as an initial input, “ball”, “squid” and “heart” were recalled correctly. In the same way, “musical note” was given, “squid”, “rabbit” and “sunflower” were recalled. In these sequential patterns, “squid” is the common term. As shown in Fig. 4, the proposed model could recall sequential analog patterns including a common term correctly.

4.2 Storage Capacity

Here, we examined the storage capacity of the proposed model. Figure 5 shows the storage capacities of the proposed model and the conventional Kohonen

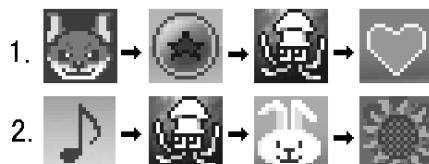
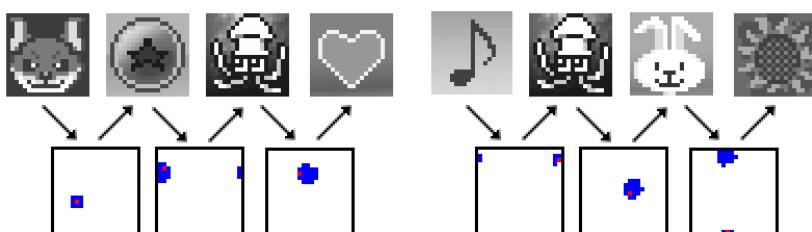
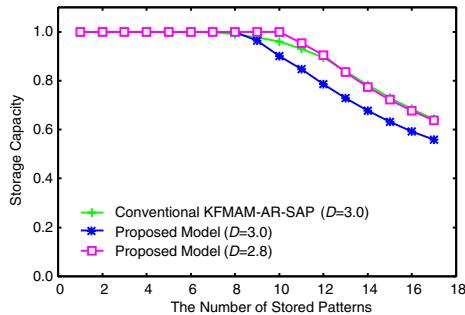


Fig. 3. Stored Sequential Analog Patterns



(a) When “fox” was given. (b) When “musical note” was given.

Fig. 4. Association Result

**Fig. 5.** Storage Capacity

Feature Map Associative Memory with Area Representation for Sequential Analog Patterns (KFMAM-AR-SAP). As shown in Fig. 5, the storage capacity of these two models are almost even. Strictly speaking, the storage capacity of the proposed model is slightly smaller than that of the conventional model. This is because the area sizes for the training patterns in the Map Layer are almost same in the proposed model. In contrast, some areas in the Map Layer are sometimes very small in the conventional model, and as a result the number of stored patterns becomes large.

4.3 Robustness for Damaged Neurons/Noisy Input

Here, we examined the robustness for damaged neurons and noisy input of the proposed model, and the conventional KFMAM-AR-SAP. In these experiments, five random pattern sequences composed of four patterns were memoized. Figure 6 shows the robustness of the proposed model and the conventional KFMAM-AR-SAP. As shown in this figure, the proposed model has enough robustness for damaged neurons and noisy input as similar as the conventional model.

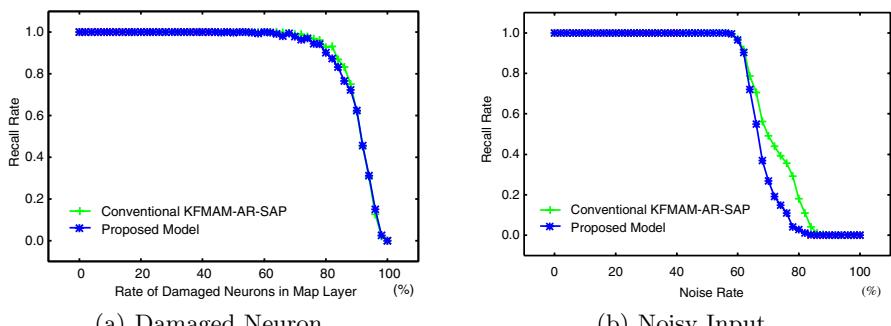
**Fig. 6.** Robustness for Damaged Neuron/Noisy Input

Table 1. Learning Time

	Learning Speed (sec)
Conventional Model	6.622
Proposed Model	0.321

4.4 Learning Speed

Here, we examined the learning speed of the proposed model. In this experiment, one random pattern sequence composed of four patterns was memorized. Table II shows the learning time of the proposed model and the conventional KFMAM-AR-SAP. These results are average of 100 trials on the Personal Computer (AMD Phenom X4 9750 (2.40GHz), Microsoft Windows Vista Home Premium 64-bit Edition, Microsoft Visual C++ 2005). As shown in Table II, the learning time of the proposed model is shorter than that of the conventional model.

5 Conclusions

In this paper, we have proposed the improved Kohonen feature map associative memory with area representation for sequential analog patterns. We carried out a series of computer experiments and confirmed that the proposed model has following features.

- (1) It can learn sequential patterns successively.
- (2) It can deal with sequential analog patterns including common terms.
- (3) Its learning speed is higher than that of the conventional KFM associative memory with area representation for sequential analog patterns.
- (4) It has large storage capacity.
- (5) It has robustness for noisy input.
- (6) It has robustness for damaged neurons.

References

1. Rumelhart, D.E., McClelland, J.L., The PDP Research Group: Parallel Distributed Processing, Exploitations in the Microstructure of Cognition. Foundations, vol. 11. The MIT Press, Cambridge (1986)
2. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1994)
3. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of National Academy Sciences USA 79, 2554–2558 (1982)
4. Kosko, B.: Bidirectional associative memories. IEEE Transactions on Neural Networks 18(1), 49–60 (1988)
5. Watanabe, M., Aihara, K., Kondo, S.: Automatic learning in chaotic neural networks. IEICE-A J78-A(6), 686–691 (1995) (in Japanese)
6. Osana, Y., Hagiwara, M.: Successive learning in chaotic neural network. International Journal of Neural Systems 9(4), 285–299 (1999)

7. Kawasaki, N., Osana, Y., Hagiwara, M.: Chaotic associative memory for successive learning using internal patterns. In: IEEE International Conference on Systems, Man and Cybernetics (2000)
8. Ideguchi, M., Sato, N., Osana, Y.: Hetero chaotic associative memory for successive learning using internal patterns. In: Proceedings of International Symposium on Nonlinear Theory and its Applications, Bruges (2005)
9. Arai, T., Osana, Y.: Hetero chaotic associative memory for successive learning with give up function — One-to-many associations. In: Proceedings of IASTED Artificial Intelligence and Applications, Innsbruck (2006)
10. Ando, M., Okuno, Y., Osana, Y.: Hetero chaotic associative memory for successive learning with multi-winners competition. In: Proceedings of IEEE International Joint Conference on Neural Networks, Vancouver (2006)
11. Ikeya, T., Sazuka, T., Hagiwara, A., Osana, Y.: Improved chaotic associative memory for successive learning. In: Proceedings of IASTED Artificial Intelligence and Applications, Innsbruck (2008)
12. Ichiki, H., Hagiwara, M., Nakagawa, M.: Kohonen feature maps as a supervised learning machine. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1944–1948 (1993)
13. Yamada, T., Hattori, M., Morisawa, M., Ito, H.: Sequential learning for associative memory using Kohonen feature map. In: IEEE and INNS International Joint Conference on Neural Networks, paper no. 555, Washington D.C. (1999)
14. Hattori, M., Arisumi, H., Ito, H.: SOM associative memory for temporal sequences. In: IEEE and INNS International Joint Conference on Neural Networks, Honolulu, pp. 950–955 (2002)
15. Abe, H., Osana, Y.: Kohonen feature map associative memory with area representation. In: Proceedings of IASTED Artificial Intelligence and Applications, Innsbruck (2006)
16. Ikeda, N., Hagiwara, M.: A proposal of novel knowledge representation (Area representation) and the implementation by neural networks. In: International Conference on Computational Intelligence and Neuroscience, vol. III, pp. 430–433 (1997)
17. Iwai, Y., Osana, Y.: Kohonen feature map associative memory with area representation for sequential patterns. In: Proceedings of International Symposium on Nonlinear Theory and its Applications, Vancouver (2007)
18. Shiratori, T., Osana, Y.: Kohonen feature map associative memory with area representation for sequential analog patterns. In: Proceedings of IEEE and INNS International Joint Conference on Neural Networks, Hong Kong (2008)

Surface Reconstruction Method Based on a Growing Self-Organizing Map

Renata L.M.E. do Rego^{1,2}, Hansenclever F. Bassani¹,
Daniel Filgueiras¹, and Aluizio F.R. Araujo¹

¹ Center of Informatics at Federal University of Pernambuco, Recife/Brazil

² Federal Institute of Pernambuco, Recife/Brazil

{rlmer, hfb, dfg, aluizioa}@cin.ufpe.br

Abstract. This work introduces a method that produces triangular mesh representation of a target object surface. The new surface reconstruction method is based on Growing Self-organizing Maps, which learns both the geometry and the topology of the input data set. Each map grows incrementally producing meshes of different resolutions, according to different application needs. Experimental results show that the proposed method can produce triangular meshes having approximately equilateral faces, that approximate very well the shape of an object, including its concave regions and holes, if any.

Keywords: Surface reconstruction, self-organizing maps.

1 Introduction

Surface reconstruction has been an important research topic due to (a) the large variety of application areas, such as medicine, cultural artifacts and robotics; (b) the recent advances in scanning technology to capture massive amounts of geometric data; (c) the innovations on hardware technology of computers allowing the visualization and manipulation of large three-dimensional data.

Surface reconstruction aims at producing a digital representation of the shape of a real world object given a set of points from its surface. A surface reconstruction method can be classified as static methods [1,2], based on geometric techniques, and dynamic methods [3,4], based on the evaluation of energy or force functions. The limitation of the static methods is that they process the points directly, and thus, can not deal with large point sets. The limitation of the dynamic methods concerns those target shapes not reachable through the modification of the initial mesh. For example, when the target shape is a torus and the initial mesh is a sphere, i.e., objects non-topologically equivalent. Surface reconstruction can also use learning-based methods which can process very large and/or noisy data, such as point clouds obtained from 3D scanners. Following this approach, some researches [5,6,7,8] employed methods based on a self-organizing map for surface reconstruction.

In this paper we propose a learning-based surface reconstruction method that is an improved version of a previous work [8]. The main limitation of [8] that

is addressed in this work concerns edges with more than two incident faces, yielding non-manifold meshes. The method proposed in this paper produces two manifold meshes. Another improvement of the method proposed here in comparison with our previous work is the addition of a condition to avoid long edges, and thus, skinny triangles. Some positive aspects of our previous work present in the method proposed here are the ability to learn different topologies from unstructured point clouds and to produce meshes with different resolutions so that a posterior step for simplifying the mesh is not necessary.

The rest of this paper is organized as follows: Section 2 presents related works using self-organizing maps for surface reconstruction. The proposed method is presented in Section 3. The experiments and results are presented in Section 4 and Section 5 concludes this paper.

2 Surface Reconstruction Methods Based on Self-Organizing Maps

Self-organizing maps are in general able to drag an initial grid of nodes towards a set of input points. For this reason, some researchers use SOM and some of its variants as the basis of surface reconstruction methods adapting an initial polygonal mesh to a given point cloud. In these works, the SOM nodes represent the vertices of the polygonal mesh; the connections between the nodes represent the edges of the mesh; and the faces of the mesh, however, do not have a direct representant in the map structure. Hereafter, the terms nodes and vertices, edges and connections, will be used interchangeably.

Some reconstruction algorithms [67] are based on the original SOM [9]. These methods are suitable to represent the faces of a polygonal mesh since the SOM structure does not change during the learning process and thus, the faces can be established in advance. The main restrictions of SOM for surface reconstruction, addressed in [67], are their fixed topology and pre-defined number of vertices.

Some SOM variants [10] are also used as the starting point for surface reconstruction methods. The method proposed in [11], based on the Topology Representing Networks (TRN) [12], can learn the topology of the input data. As the TRN topological structure changes during the learning process, the faces of the desired polygonal mesh representation cannot be determined in advance as in the original SOM based methods. For this reason the surface reconstruction method proposed in [11] extends TRN to define faces. TRN has a pre-defined number of nodes, hence the meshes produced by TRN have a resolution dependent on the pre-defined number of vertices. Furthermore, the topology learning strategy of TRN does not produce a two-manifold mesh. To solve this limitation, Barhak [11] included a post-processing step responsible for creating manifold meshes.

The Neural Meshes [13], another learning based surface reconstruction method, employs the Growing Cell Structures (GCS) [14]. The structure of GCS map consists of k -dimensional simplices [14], for $k = 2$ triangles represent the faces of a triangle mesh. GCS maps grow incrementally producing meshes with different resolutions, avoiding mesh simplification. A relevant restriction of the standard

GCS for reconstructing surfaces is when the target shape is not topologically equivalent to the initial mesh.

The Growing Neural Gas (GNG) can learn the geometry and topology of an input point cloud, and it grows incrementally. Thus, we propose a method having GNG as its starting point.

3 Self-Organizing Solution Proposed for the Surface Reconstruction Problem

The proposed method receives as input the 3D coordinates of a set of points randomly sampled from the surface of a target object, learns its geometry and topology, and outputs triangular meshes representation of the target object surface with different resolutions. The neural solution proposed here addresses some challenges of the surface reconstruction task:

- Production of 2-manifold triangular mesh approximation of the target surface.
- Reconstruction of meshes according to different resolutions.
- Learning of the geometry (vertex coordinates) and topology of the mesh.
- Learning of the topology (vertex connectivity) of the mesh without any structural information.
- Reconstruction of surfaces with different topologies.

The proposed reconstruction method, called Growing Self-reconstruction Maps (GSRM), is a modified version of GNG aiming to support the representation of triangular meshes, to generate 2-manifold meshes, and to avoid long edges to produce only triangles which are approximately equilateral. The differences between GSRM and the standard GNG concern the Competitive Hebbian Learning algorithm (Section 3.2), the procedure for edge removal (Section 3.3), and the vertex insertion operator (Section 3.4). The step-by-step runthrough of the GSRM learning algorithm is presented in Section 3.1.

3.1 GSRM Learning Algorithm

The GSRM learning algorithm relies on six parameters: ε_b and ε_n - learning rate of the winner node and its neighbours, respectively ($\varepsilon_b > \varepsilon_n$); λ - frequency at which a new node is inserted; α - error reduction rate of the nodes that are neighbours of a node that has just been inserted; β - error reduction rate that aims at stressing the impact of recently accumulated errors ($\beta < \alpha$); age_{max} - maximum age for an edge to be removed.

The input for the GSRM learning algorithm is a point cloud P . The GSRM learning algorithm is described below:

1. Initialize the map (A) with three nodes. The weight vector of these nodes are randomly chosen from P .
2. Present a sample ξ , randomly chosen from P .

3. Find the two nodes (s_1 , s_2) of the map that are nearest to ξ according to the Euclidian distance.
4. If a connection between s_1 and s_2 does not exists:
 - Then, create such a connection (e), and create faces incident to e , according to ECHL (Section 3.2).
 - Else, reinforce the edge e connecting s_1 and s_2 ($age_e = 0$) and check the other edges of the mesh against a condition for edge removal based on the Thales Sphere concept (Section 3.3).
5. Update the error counter of node s_1 :

$$\Delta E = \|\mathbf{w}_{s_1} - \xi\|^2 \quad (1)$$

where \mathbf{w}_{s_1} is the weight vector of node s_1 .

6. Adapt the weight vector of node s_1 and its neighbors.

$$\Delta w_{s_1} = \varepsilon_b * (\xi - \mathbf{w}_{s_1}) \quad (2)$$

$$\Delta w_{s_n} = \varepsilon_n * (\xi - \mathbf{w}_{s_n}) \quad \forall s_n \in N_i \quad (3)$$

where N_i is the neighborhood of node s_1 .

7. Update the age of all edges e emanating from s_1 .

$$age_e = age_e + 1 \quad (4)$$

8. Remove the faces coincident to an old edge e ($age_e > age_{max}$) and remove this edge.
9. If the number of samples presented so far is greater than λ , insert a new node in the map according to the procedure presented in Section 3.4.
10. Decrease the error variables of all nodes:

$$\Delta E_s = -\beta E_s \quad \forall s \in A \quad (5)$$

11. If the map achieved the desired resolution, complete the topological learning and perform the post processing step according to the procedure presented in Section 3.5.

3.2 Extended Competitive Hebbian Learning (ECHL)

An extended version of the Competitive Hebbian Learning rule (CHL) [15] is proposed for the creation of edges and faces. The two main modifications are: (a) before connecting two nodes (s_1 and s_2), the extended CHL checks if s_1 and s_2 have two connected common neighbors (n_1 and n_2), and if this is true, the edge connecting n_1 and n_2 is removed before creating the new edge connecting s_1 and s_2 , to avoid overlapping edges; and (b) beyond edges, faces are also created which are formed by the nodes s_1 and s_2 , and their common neighbors, if any.

Before creating a new face, the following conditions must be satisfied to avoid more than two faces sharing the same edge, and thus non 2-manifold meshes: (a) s_1 and s_2 must have at most two common neighbors, otherwise no face is created; (b) an edge e connecting s_1 or s_2 to a common neighbor must have at most one coincident face before creating a new face inciding on e , otherwise such a new face is not created.

The CHL extension just presented is based on the ideas of [11], however, we have introduced conditions to avoid overlapping edges and to produce 2-manifold meshes.

3.3 Edges and Incident Faces Removal

Some of the edges generated in the network may become invalid, then, they have to be removed. In this work, two mechanisms are used to identify such edges. The first one, employed by the standard GNG, consists of an edge ageing scheme. The second mechanism removes long edges to create triangular faces that are approximately equilateral. It is based on the Thales Sphere concept and has been presented in a related work [16]. The following procedure is used to identify long edges: If the angle σ between the vectors $\mathbf{v} = \mathbf{w}_{s_1} - \mathbf{w}_{s_2}$ and $\mathbf{u} = \mathbf{w}_{s_k} - \mathbf{w}_{s_2}$ (\mathbf{w}_{s_i} is the weight vector of node s_i ; s_1 is the winner node, s_2 is the second winner, and s_k is a neighbor of the winner node), is greater than $\frac{\pi}{2}$, then the edge connecting s_1 and s_k is removed.

This condition is verified whenever the connection to be created by a sample presentation already exists (step 4 of the algorithm presented in Section 3.1).

GNG simply removes an invalid edge and the vertices without any incident edges. The edge removal operation in GSRM takes face definition into account and removes the faces having an invalid edge before removing this edge. If the edge removal yields vertices without incident edges, these vertices are also removed.

3.4 Vertex Insertion

In standard GNG, a new vertex (s_r) is inserted between two existing vertices (s_q and s_f , where s_q is the vertex with highest accumulated error, and s_f is the neighbor of s_q with highest accumulated error). The original edge connecting s_q and s_f is removed and two new edges, connecting s_q and s_f to s_r , are then created. GSRM considers face definition and, before removing the original edge, it removes each face inciding on this edge.

3.5 Topological Learning and Post-processing Step

At the end of the learning process the algorithm has learned the vertex coordinates and some of the edges and faces of the mesh. However some edges needed for the representation may not have been created and some long edges may not have been removed due to the non-deterministic behavior of the learning process. To overcome these consequences we complete the topological learning after the

mesh resolution desired is reached. The topological learning consists on removing long edges, according to the rules discussed in Subsection 3.3 and presenting all of the samples, to create all the necessary edges, according to the extended CHL presented in Subsection 3.2.

After the topological learning, some approximately regular polygons remain untriangulated after the learning process is finished. This occurs because any sample internal to those polygons has two connected nodes as winners, as illustrated in Figure 1(a), then CHL is unable to complete the triangulation internal to the polygon [16]. In order to complete the triangulation a new vertex is inserted in the geometric center of each polygon and the triangulation is possible by connecting this new vertex to each of the polygons vertices, see Figure 1(b) for an example. When the polygon is a quadrilateral, the triangulation is performed by creating its diagonal edge and replacing it by two triangles.

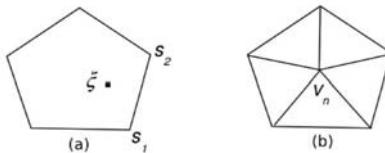


Fig. 1. Triangulating a polygon with five vertices

4 Experiments and Results

This section presents the experimental results carried out by the proposed surface reconstruction method. We first present visual results of surface reconstruction of synthetic objects. Then, we present numeric metrics to evaluate the reconstructed meshes and to compare our results with those of some related works.

The input point cloud has been acquired by randomly sampling points from the target synthetic object surface. The values of the parameters used in the experiments presented in this paper are: $\varepsilon_b = 0.05$, $\varepsilon_n = 0.0006$, $\lambda = 200$, $\alpha = 0.5$, $\beta = 0.0005$, $age_{max} = 30$. These values have been chosen by trial and error. Initial values were those presented in [10]. Then, these initial values have been gradually adjusted to better fit the proposed learning method.

The synthetic objects reconstructed are the hand and the Max-Planck, donated by Ioannis Ivrissmitzis [13], and the bunny, available at the Stanford repository (<http://www-graphics.stanford.edu/data/3Dscanrep/>). Figure 2 presents the reconstructions outputted by the proposed surface reconstruction method. All the reconstructions presented have about 20,000 vertices.

An important remark is the topology learning ability of the proposed surface reconstruction method. Note, for example, the hole boundaries reproduced in the bottom of the hand and the bunny (Figure 2). Note also that concave and detailed regions can be reproduced, such as the space between the fingers of the hand, and the ears of the bunny and the relief on its body.

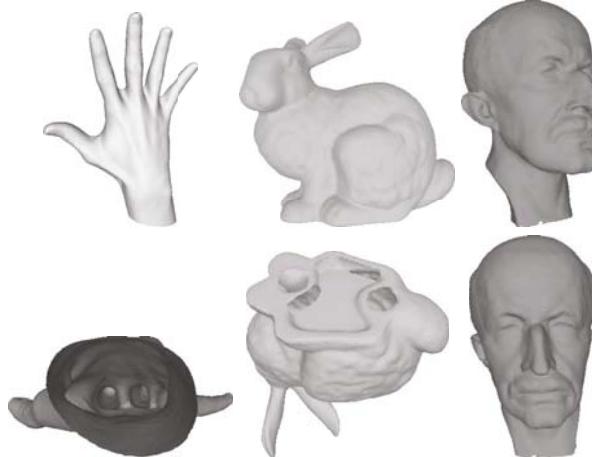


Fig. 2. Screenshots of GSRM Reconstructions

4.1 Evaluation Metrics of Reconstructed Meshes

The metrics used for evaluating the reconstructed meshes are: the distance between the target and the reconstructed surfaces, polygon conformity, and valence distribution. To get a more reliable estimate on the performance of the proposed algorithm, the values of the metrics presented here represents the average of the values observed in three runs, since the proposed algorithm is a stochastic one.

The distance between the target surface and the reconstructions are given in terms of Hausdorff distance, defined as follows: Given a point p and a surface S , the distance $e(p, S)$ is defined as:

$$e(p, S) = \min d(p, p') \quad \forall p' \in S \quad (6)$$

where $d()$ is the Euclidean distance between two points.

The one-sided distance between two surfaces S_1, S_2 is then defined as:

$$E(S_1, S_2) = \max e(p, S_2) \quad \forall p \in S_1. \quad (7)$$

The above definition of distance is not symmetric. There exist surfaces such that $E(S_1, S_2)$ is different from $E(S_2, S_1)$. Thus, a two-sided distance (Hausdorff distance) may be obtained by taking the maximum of $E(S_1, S_2)$ and $E(S_2, S_1)$. Note that, the smaller the Hausdorff distance, the better the reconstruction approximates the target shape.

Polygon conformity $R(P)$ is measured as the ratio between the smallest and the largest distance of the vertices of the polygon (s) to the polygon baricenter (b_p), see Equation (8). If the polygon tends to be elongated, $R(P)$ approximates to zero, but if the polygon tends to be regular, $R(P)$ is close to 1. Note that a triangle tends to be equilateral when its $R(P)$ value tends to 1.

$$R(P) = \frac{\min\|s - b_p\|}{\max\|s - b_p\|}, \quad (8)$$

Valence distribution refers to the number of neighbors of the mesh vertices. Valences must be distributed as evenly as possible [13], this means that the vertices of the mesh must have approximately the same number of neighbors.

It is important to compare the reconstructions of the proposed method with the reconstructions of other surface reconstruction methods: two versions of the Neural Meshes [17], the SOM based surface reconstruction [6], and the Power Crust [18], a traditional method. The values of the metrics related to the other methods, have been taken from their original papers. However, the original papers do not present values for all the metrics used here. For example, [13] presents the valence distribution of the meshes, but it does not present polygon conformity and Hausdorff distances. Thus, the comparisons presented here are according to the data available in the related papers.

Table 1 presents the Hausdorff distance from the original synthetic object to the reconstructions produced by GSRM and by two versions of the Neural Meshes algorithm [17]. Both the Neural Meshes and GSRM reconstructions have about 20,000 vertices. Neural Meshes performances were taken from [17]. GSRM reconstructions reproduce the shape of the original objects better than the Neural Meshes reconstructions. The GSRM reconstructions smaller distances from the original objects is due to the boundaries of the original objects (see the bottom of the bunny and the hand in Figure 2) that are reproduced in the GSRM reconstructions but not in the Neural Meshes reconstructions.

Table 2 compares the polygon conformity of the GSRM bunny (three options) with the Power Crust [18] and the SOM-based [6] reconstructions. The values for the reconstructions of the two last methods were extracted from [6]. From the values presented, GSRM reconstructions has better polygon conformity even with less elements (vertices and faces).

The valence distributions of GSRM and Neural Meshes reconstructions are shown in Table 3. The values for the Neural Meshes reconstructions were extracted from [13]. The first column of Table 3 refers to the name of the model and the approximate number of vertices. According to Table 3, most of the vertices have about five to seven neighbors both in the reconstructions produced by GSRM and Neural Meshes. Thus, both methods distribute valences almost evenly. The valence distribution of the meshes produced by the Neural Meshes, however, is slightly better.

Table 1. Hausdorff distances between original meshes and the reconstructions produced by two versions of Neural Meshes GSRM

	Max-Planck	Hand	Bunny
Neural Mesh I	5.23	6.60	0.006
Neural Mesh II	3.86	8.30	0.007
GSRM	0.002602	0.0013207	0.001513

Table 2. Polygon conformity comparison

Method	No. of vertices	No. of faces	$R(P)$
Power Crust	277,242	191,981	0.519424
SOM	31,004	62,004	0.5676
GSRM	20,108	39,995	0.6886
GSRM	5,006	9,906	0.6893
GSRM	1,010	1,978	0.6855

Table 3. Valence distribution comparison

Model	Valence Distribution(%)											
	GSRM						Neural Meshes					
	4	5	6	7	8	others	4	5	6	7	8	others
Sphere 1k	4.13	24.90	43.81	22.67	3.96	0.53	0.20	29.60	47.00	18.40	4.40	0.50
Bunny 1k	5.89	26.48	39.60	21.20	5.17	2.09	0.70	27.40	49.00	18.60	4.20	0.20
Sphere 5k	4.12	26.15	41.61	22.68	4.87	0.56	0.20	29.02	46.60	19.98	3.54	0.68
Bunny 5k	5.00	26.28	40.87	22.24	4.60	1.00	0.44	28.32	47.28	19.74	3.54	0.70

5 Conclusions

The neural method put forward in this article succeeded in reconstructing surface models of 3D objects from point clouds representing their shape. The highlights of the proposed method are (i) topology learning, which is a challenging feature for reconstruction methods, (ii) models generated at different resolutions, (iii) meshes with consistent polygon conformity and valence distribution.

At the moment, an important limitation of the proposed method concerns reconstruction time, which strongly depends on the number of nodes in the map. The main reason for this dependency is the linear search for winner nodes used in step 3, and the linear search for the node with highest accumulated error used in the step 9 of the proposed algorithm. To solve this limitation we propose, as a future work, implementing a more efficient search scheme in both situations.

References

1. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: Proceedings of Siggraph Conference, pp. 71–78 (1992)
2. Amenta, N., Bern, M., Kamvysselis, M.: A new voronoi-based surface reconstruction algorithm. In: Siggraph Conference Proceedings, pp. 415–422 (1998)
3. Miller, J., Breen, D., Lorensen, W., O’Bara, R., Wozny, M.: Geometrically deformed models: A method for extracting closed geometric models from volume data. In: Proceedings of the International Conference on Computer Graphics and Interactive Techniques, vol. 25, pp. 217–226 (1991)
4. Qin, H., Mandal, C., Vemuri, B.: Dynamic catmull-clark subdivision surfaces. IEEE Transactions on Visualization and Computer Graphics 4(3), 215–229 (1998)

5. Hoffmann, M., Varady, L.: Free-form modelling surfaces for scattered data by neural networks. *Journal for Geometry and Graphics* 2(1), 1–6 (1998)
6. Brito, A., Doria, A., de Melo, J., Goncalves, L.: An adaptive learning approach for 3d surface reconstruction from point clouds. *IEEE Transactions on Neural Networks* 19(6), 1130–1140 (2008)
7. Yu, Y.: Surface reconstruction from unorganized points using self-organizing neural networks. In: *Proceedings of IEEE Visualization Conference*, pp. 61–64 (1999)
8. do Rego, R.L.M.E., Araujo, A.F.R., de Lima Neto, F.B.: Growing self-organizing maps for surface reconstruction from unstructured point clouds. In: *International Joint Conference on Neural Networks*, pp. 1900–1905 (2007)
9. Kohonen, T.: *Self-Organizing Maps*. Springer, Berlin (2001)
10. Fritzke, B.: Unsupervised ontogenetic networks. *Handbook of Neural Computation* (1996)
11. Barhak, J.: Freeform objects with arbitrary topology from multirange images. PhD thesis, Institute of Technology, Haifa, Israel (2002)
12. Martinetz, T., Schulten, K.: Topology representing networks. *Neural Netw.* 7(3), 507–522 (1994)
13. Ivrissimtzis, I., Jeong, W.K., Seidel, H.P.: Using growing cell structures for surface reconstruction. In: *Proceedings of International Conference on Adaptive and Natural Computing Algorithms*, pp. 78–86 (2003)
14. Fritzke, B.: Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks* 7, 9–13 (1994)
15. Martinetz, T.M., Schulten, K.J.: A neural-gas network learns topologies. *Artificial Neural Networks*, 397–402 (1991)
16. DalleMole, V., Araujo, A.F.R.: The growing self-organizing surface map. In: *International Joint Conference on Neural Networks*, pp. 2061–2068 (2008)
17. Saleem, W.: A flexible framework for learning-based surface reconstruction. Master's thesis, Computer Science Departament, University of Saarland, Saabracken, Germany (2003)
18. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pp. 249–266 (2001)

Micro-SOM: A Linear-Time Multivariate Microaggregation Algorithm Based on Self-Organizing Maps

Agusti Solanas, Arnau Gavalda, and Robert Rallo

Department of Computer Engineering and Mathematics,

Rovira i Virgili University,

Av.Països Catalans, 26,

43007 Tarragona, Catalonia, Spain

{agusti.solanas,arnau.gavalda,robert.rallo}@urv.cat

Abstract. The protection of personal privacy is paramount, and consequently many efforts have been devoted to the study of data protection techniques. Governments, statistical agencies and corporations must protect the privacy of the individuals while guaranteeing the right of the society to knowledge. Microaggregation is one of the most promising solutions to deal with this praiseworthy task. However, its high computational cost prevents its use with large amounts of data. In this article we propose a new microaggregation algorithm that uses self-organizing maps to scale down the computational costs while maintaining a reasonable loss of information.

Keywords: Self-Organizing Maps, Privacy, k -Anonymity, Microaggregation.

1 Introduction

The Information and Communications Technologies (ICT) pave the way for the storage and analysis of huge amounts of data. It is paramount that the collection and, specifically, the analysis of all these data take into account the privacy of individuals. Hence, there must be a balance between the right of the society to information and the right of individuals to privacy. Some common examples of information gathering and analysis using ICT are the following: (i) **E-commerce**: Commercial information that is collected from fidelity cards, e-commerce logs, sales software, etc. From this information, customer's habits can be inferred, so these data analysis can feed marketing strategies; (ii) **Statistical Data**: Statistical agencies collect data from individuals to analyze them and publish statistical reports. These data, which are kept in large databases, might be released to third parties such as marketing companies and, as a result, statistical agencies lose control on the data once they are released; (iii) **Location Information**: Location-based services use the location of the users to provide them with personalized services. These locations could be used to infer the habits of

the users, thus, these data must be managed carefully; and (iv) **Internet search engines:** Internet advertising companies use browsing information of Internet users to provide them with personalized advertisements. This practice could be seen as a privacy invasion if users are not properly informed.

These examples above are just the tip of the iceberg. ICT users have to face these situations daily and, although privacy invasion is becoming a general practice, privacy is a fundamental right¹ that must be protected; Governments, public institutions and corporations must provide the means to do so.

Due to the importance of individual privacy protection, several techniques have been proposed to cope with this problem, namely noise addition [1], rank swapping [2], statistical obfuscation [3], microaggregation [4], etc. A more general and comprehensive survey on security-control techniques for statistical databases can be found in [5]. To the best of our knowledge, microaggregation is one of the most promising techniques for microdata protection because it achieves a good balance between information loss and disclosure risk. Unfortunately, microaggregation has a high computational cost (i.e. $O(n^2)$) that prevents its use on large data sets.

In this article, we propose a new linear-cost microaggregation algorithm (i.e. $O(n)$) based on self-organizing maps (SOM). Although microaggregation and SOM are well-known techniques in the fields of statistical disclosure control (SDC) and artificial intelligence respectively, this is the first time in which they are considered together to protect individual privacy.

The rest of the article is organized as follows: In Section 2 we provide the reader with some background on microaggregation and self-organizing maps. Our linear-time microaggregation algorithm is described in Section 3 and, the experimental results are shown in Section 4. Finally, in Section 5 we conclude the article with some final comments and future research lines.

2 Background

In this section we provide the reader with some basic concepts on microaggregation and k -anonymity (Section 2.1), and self-organizing maps (Section 2.2).

2.1 Microaggregation and k -Anonymity

k -Anonymity is an interesting approach to face the conflict between information loss and disclosure risk, suggested by Samarati and Sweeney [6,7].

Definition. *A protected data set is said to satisfy k -anonymity for $k > 1$ if, for each combination of attributes, at least k records exist in the data set sharing that combination.*

¹ “No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor or reputation . . .” *Universal Declaration of Human Rights*.

The original computational approach to achieve k -anonymity relied on suppressions and generalizations, so that minimizing information loss translates to reducing the number and/or the magnitude of suppressions and generalizations. Unfortunately, the original k -anonymity approach has several drawbacks when applied to numerical data, being the most relevant one the categorization of the numerical non-categorical data. It has been shown that k -anonymity can be achieved without categorization on numerical data by means of microaggregation [4].

Microaggregation is a statistical disclosure control (SDC) sub-discipline devoted to the protection of individual numerical data, also called *microdata*. It can be understood as a clustering problem. However, instead of considering the number of clusters as the main constraint, microaggregation is constrained by the size of the clusters. The microaggregation problem can be stated as follow:

Given a data set \mathbf{D} with n records in a characteristic space \mathbb{R}^d , the problem consists in obtaining a k -partition² \mathcal{P} of \mathbf{D} , so that the sum of squared errors in each part of \mathcal{P} is minimized. Once \mathcal{P} is obtained, each record of every part of \mathcal{P} is replaced by the average record of the part.

In order to determine the information loss produced by microaggregation the sum of squared errors (SSE) is used (*cf.* Expression 1).

$$\text{SSE} = \sum_{i=1}^g \sum_{j=1}^{n_i} (\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)' (\mathbf{x}_{ij} - \bar{\mathbf{x}}_i) \quad (1)$$

where g is the number of groups/parts, n_i is the number of records in the i -th group/part, \mathbf{x}_{ij} is the j -th record in the i -th group/part and $\bar{\mathbf{x}}_i$ is the average record of the i -th group/part.

The SSE is generally compared with the total error (SST) defined in Equation 2 to obtain a measure of the information loss.

$$\text{SST} = \sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})' (\mathbf{x}_i - \bar{\mathbf{x}}) \quad (2)$$

where n is the number of records in the data set, x_i is a record of the data set and \bar{x} is the average record of the data set.

The microaggregation problem is known to be NP-hard [8] for multivariate data sets, therefore heuristic methods should be used to solve it. There is a plethora of methods to address the multivariate microaggregation problem. Some of them are based on building tree structures connecting the records in the data set and partition the tree to generate a k -partition [4][9]. Their main limitation is the high computational cost due to the computation of the distances between *all* pairs of records in the data set.

Instead of structuring the data in trees or graphs, an alternative way to tackle the problem is to build groups of similar records greedily. Examples of this approach are the Maximum Distance to Average Vector (MDAV) method [10], and

² A k -partition of \mathbf{D} is a partition where its parts have, at least, k records of \mathbf{D} .

the Variable Maximum Distance to Average Vector (V-MDAV) method [11]. The main advantage of these approaches is their simplicity but their computational cost (*i.e.* $O(n^2)$) prevents their use with very large data sets.

Recent articles have tackled the problem of improving the k -partition obtained by previous microaggregation methods with the aim to reduce the information loss (but not the computational cost) [12][13]. On the other hand, microaggregation methods with low computational costs have been barely studied. A very recent example on this line can be found in [14], where the authors propose a linear-time microaggregation algorithm that works on the assumption that the input data are uniformly distributed.

The reduction of the computational cost of microaggregation algorithms in which the distribution of the input data is unknown is a key problem that we address in this article.

2.2 Self-Organizing Maps

The Self-Organizing Map (SOM) algorithm is based on an unsupervised competitive learning approach. The training process is entirely data-driven and map units compete to become specific detectors of certain data features. Each map unit is represented by an n -dimensional weight vector, where n is equal to the dimension of the input space. As in vector quantization, every weight vector describing a class is called a codebook. Each unit i has a topological neighborhood N_i determined by the shape of the SOM grid lattice which can be either rectangular or hexagonal. The number of units as well as their topological relations are defined during the initial map formation phase. The granularity (*i.e.* the size) of the map determines its accuracy and generalization capabilities. The number of units should usually be selected to accommodate all training data, with the neighborhood size controlling the smoothness and generalization of the mapping. The use of a hexagonal lattice is usually recommended, because all six neighbors of a unit are at the same distance, as opposed to the eight neighbors in a rectangular lattice configuration. The shape of the map grid should correspond to the shape of the data manifold whenever possible. To avoid the border effects in the mapping process, *i.e.*, units with a reduced neighborhood, a periodic shape such as a torus is used. Additional details about the algorithm and its implementation can be found in [15]. The quality of a SOM can be evaluated from the resolution of the map and from the preservation of the topology of the native data set. The most important issue regarding the accuracy of the SOM projection is the “true” dimension of data. If it is larger than the dimension of the map grid the SOM may not follow the distribution of the data set. In this case topology preservation and map resolution become contradictory goals and cannot be simultaneously attained. When this occurs, a map with high resolution folds into itself and topology is broken. SOM resolution is measured by means of the average quantization error over the complete data set. SOM quality can also be estimated with a combined strategy that mixes both topology and resolution [16]. Once the training completes, the SOM approaches the clustering structure of the data. Clusters are formed by groups of codebook vectors which are close to

each other compared with their distance to other vectors. The clustering structure of the input space can be visualized over the SOM grid by displaying the distances between these reference vectors. Several methods have been proposed to display the clustering structure; the most common is the unified distance matrix (*U-matrix*), i.e., the matrix of distances between each codebook vector and its neighbors. The visualization of these clusters could be enhanced by labeling the map with auxiliary data.

Since it is difficult to detect clusters by visual inspection of the U-matrix, SOM's reference vectors can in turn be clustered to detect coherent sets of units with similar structural characteristics. Simpler clustering algorithms such as the K-means procedure are used to cluster SOM vectors. The clustering parameters are optimized using some clustering quality criteria such as the minimization of the Davies-Bouldin index. This index is a function of the ratio between the sum of cluster compactness and inter-cluster separations [17] and permits the selection of the optimal number of clusters to obtain a good partitioning.

There exist many other variants and applications of the basic SOM reported in the literature. For applications see e.g. [18]. Possible variations include the use of neuron specific learning rates and neighborhood sizes, and growing map structures. The goal of all these variations is to enable the SOM to follow the topology of the underlying data set better and to achieve good quantization results [19]. The *Tree Structured SOM* [20] is a fast version of the SOM that consists in a set of layers that perform a complete quantization of the data space. Data from upper layers is used to train lower layers reducing the amount of distance calculations needed to find the winner unit. The *Minimum Spanning Tree SOM* [21] uses a tree structure as neighborhood function which defines the minimal set of connections needed to link together a related set of codebook vectors.

3 Our Proposal: Micro-SOM

Given an input data set \mathbf{D} with n records, we want to obtain a partition \mathcal{P} of \mathbf{D} so that each part of \mathcal{P} has, at least, k records (i.e. a k -partition). After determining \mathcal{P} we replace each record $r_i \in \mathbf{D}$ by the centroid of the part to which it belongs, thus obtaining a k -anonymous microaggregated data set $\bar{\mathbf{D}}$.

Our algorithm, called Micro-SOM, can be divided in the following three steps:

- **STEP 1 – Obtain a partition of \mathbf{D} :** Previous methods compute the distance between all pairs of records (i.e. n^2 distances) so that they can heuristically determine subsets of \mathbf{D} built of relatively close records (cf. Section 2.1 for further details and references). Instead of computing the n^2 distances, we use a self-organized map to group records in rough subsets (by doing so, we reduce the computational cost from $O(n^2)$ to $O(n)$). Note that the number of units/neurons is clearly smaller than the number of records. In this step, the algorithm proceeds as follows:

- **STEP 1.1 – Standardize \mathbf{D} :** In order to avert the influence of the difference between the scale of the attributes in the data set, it must be standardized before training the SOM (See Figure II-a).

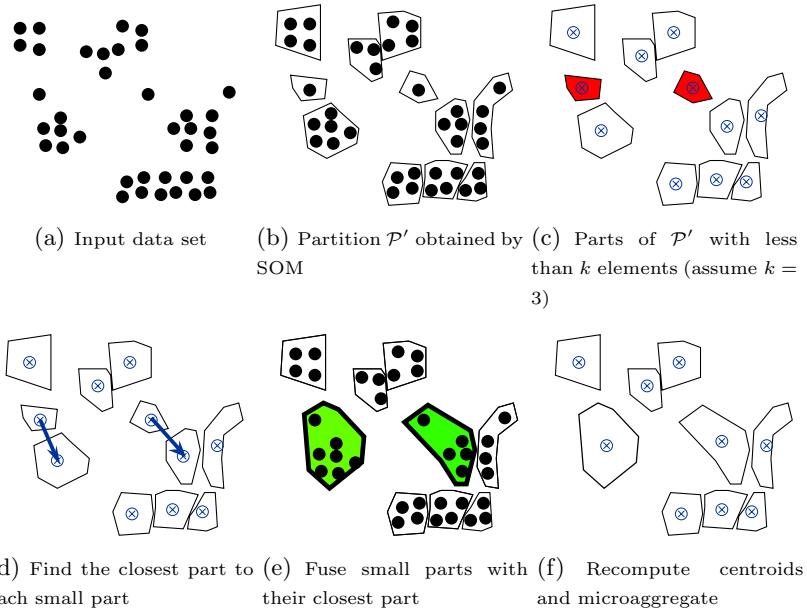


Fig. 1. Graphical representation of different steps of the proposed algorithm

- **STEP 1.2 – Initialize and train the SOM:** The size of the map is heuristically determined by using the algorithm defined in [22]. By doing so, it can be guaranteed that the number of neurons in the map is some orders of magnitude smaller than the number of records (n). Also the dimensionality of the data is some orders of magnitude smaller than the number of records. Note that thanks to these differences in the orders of magnitude the computational cost of the algorithm is linear (i.e. $O(n)$) instead of quadratic (i.e. $O(n^2)$) [23][24]. Once the SOM is initialized, it is trained by using the standardized data.
- **STEP 1.3 – Assign each record to a map unit:** After training the SOM, we can assign each record in \mathbf{D} to a map unit. Thus, the set of non-empty units represents a partition \mathcal{P}' of \mathbf{D} (See Figure 1-b).
- **STEP 2 – Obtain a k -partition:** In the previous step we get a partition, but some of the parts may have less than k elements. If that happens, low-cardinality parts must be fused with other parts so as to obtain a k -partition. To do so, the algorithm proceeds as follows:
 - **STEP 2.1 – Find low-cardinality parts:** All parts with less than k records are labelled as “low-cardinality parts” (See Figure 1-c). If there are low-cardinality parts then go to step 2.2, otherwise go to step 3.
 - **STEP 2.2 – Compute centroids and find the closest part to each low-cardinality part:** For each part, its centroid is computed. If there are low-cardinality parts, the closest part to each low-cardinality part is found by comparing the distances to its neighbors. Finding the closest

part is easy and computationally cheap due to the topological properties of SOM. (See Figure II-d)

- **STEP 2.3 – Fuse each low-cardinality part with its closest part:** Using the information obtained in the previous step, low-cardinality parts are fused with their closest parts (See Figure II-e). Before moving to the step 3, the algorithms goes back to step 2.1 to check whether low-cardinality parts still exist.
- **STEP 3 – Generate a microaggregated data set:** When we reach this step each record in the data set belongs to a part having at least k records. Also, the centroid of each part has been computed so that it can be used to replace the records belonging to that part. By replacing all records in \mathbf{D} by the centroid of the parts to which they belong, a k -anonymous microaggregated data set $\bar{\mathbf{D}}$ is obtained and the algorithm terminates (See Figure II-f).

4 Results

In order to test our solution, we have used the SOM Toolbox [22] under Matlab to implement the first step of the algorithm (i.e. Obtain a partition of \mathbf{D}). The second and third steps (i.e. Obtain a k -partition and generate a microaggregated data set) have been implemented using Java. All experiments have been carried out on an Intel Core2 Duo at 1.4Ghz with 2 GB of RAM running Windows XP.

With the aim to emphasize the low computational cost of the proposed algorithm, we have compared it with the μ -Approx algorithm [4] that is one of the most recent microaggregation algorithms proposed in the literature. We have used three real-life data sets that were used in the European CASC project [25] and two synthetic data sets with a larger number of records:

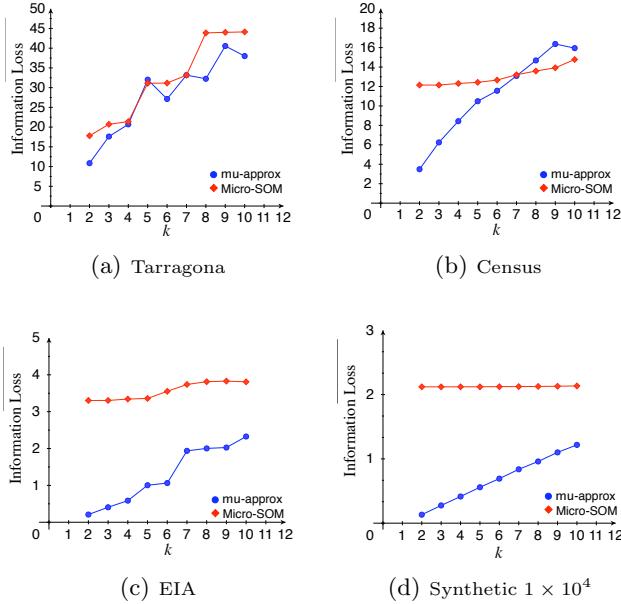
- The “Tarragona” data set contains 834 records with 13 numerical attributes.
- The “Census” data set contains 1080 records with 13 numerical attributes.
- The “EIA” data set contains 4092 records with 11 numerical attributes.
- The “Synthetic 1×10^4 ” data set contains 10.000 records with 3 numerical attributes uniformly distributed.
- The “Synthetic 1×10^5 ” data set contains 100.000 records with 3 numerical attributes uniformly distributed.

4.1 Information loss

The information loss (IL) is a widely used measure to determine how good a microaggregation algorithm is. The lower the IL the better the algorithm. Typically, the IL is defined as the ratio between SSE and SST in % (see Equation 3).

$$IL = 100 \times \frac{SSE}{SST} \quad (3)$$

We have studied the IL for the Tarragona, Census, EIA and Synthetic 1×10^4 data sets varying the minimum cardinality parameter k from 2 to 10, which

**Fig. 2.** Information loss for Micro-SOM and μ -Approx

are common values for k . The results are shown in Figure 2. From this figure, it is apparent that the behavior of both algorithms is pretty similar. Although μ -Approx achieves better results than Micro-SOM in general, the differences between them are very little specially when k grows. It is also clear that Micro-SOM is more resilient to the increase of the minimum cardinality parameter k , this is, it is more stable.

4.2 Time

The time required to microaggregate a data set is very relevant, specially when large data sets have to be managed. We have studied the time required to microaggregate the Tarragona, Census, EIA and Synthetic1 $\times 10^4$ data sets varying the parameter k from 2 to 10, which are common values for k . The results are shown in Figure 3. From this figure we can conclude the following:

- When the number of records is small enough, μ -Approx performs very similar to Micro-SOM. See, for example, the results for the Tarragona data set (843 records) and the Census data set (1080 records).
- When the number of records in the data set grows, the performance of the μ -Approx algorithm degrades significantly while Micro-SOM properly copes with the microaggregation. See, for example, the results for the EIA data set (4092 records) and the Synthetic1 $\times 10^4$ data set (1×10^4 records).

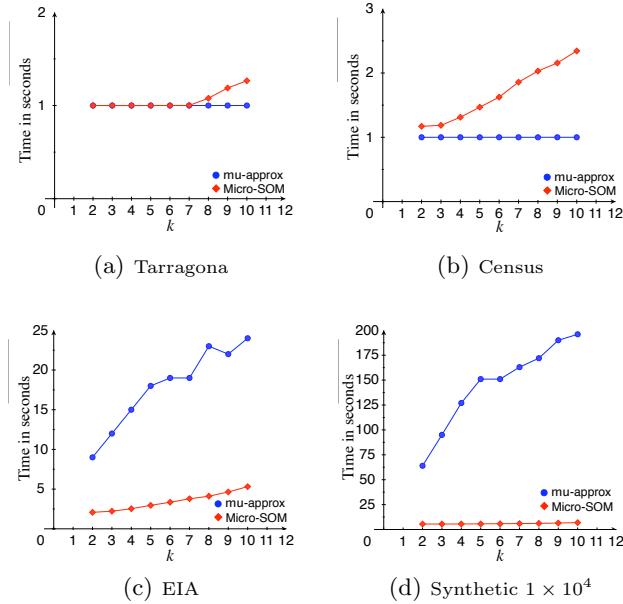


Fig. 3. Computation time for Micro-SOM and μ -Approx

As it is shown in the next section, if the number of records grows enough, the μ -Approx algorithm is unable to microaggregate them due to its quadratic (i.e. $O(n^2)$) computational cost.

4.3 Results with Large Data Sets

In the previous sections we have compared the μ -Approx algorithm with our proposal. Unfortunately, the μ -Approx algorithm cannot manage a data set consisting of 1×10^5 records because it cannot cope with a matrix of 1×10^{10} distances (i.e. the computer runs out of memory). Consequently, we only summarize in Table II the results that Micro-SOM has obtained.

Table 1. Synthetic 1×10^5

k	2	3	4	5	6	7	8	9	10
SSE	7632.0	7632.3	7632.7	7633.0	7633.5	7632.7	7631.8	7631.5	7631.9
SST	3×10^5								
IL	2.544	2.544	2.544	2.544	2.544	2.544	2.544	2.544	2.544
Time (s)	96.406	101.812	110.953	119.328	131.468	149.844	163.235	180.906	203.438

5 Conclusions and Further Work

We have presented Micro-SOM, a new microaggregation algorithm that uses self-organized maps to scale down the computational costs related to the computation of distances between all pairs of records in a data set. To the best of our knowledge, this is the first time in which a microaggregation method is fused with an artificial neural network.

The obtained results show that the information loss of Micro-SOM is similar to the information loss of μ -Approx that is one of the best microaggregation methods in the literature. The main contribution of Micro-SOM is its low-computational cost. From the results given in the previous section, it is apparent that Micro-SOM scales linearly with the number of elements in the data set and clearly outperforms the μ -Approx algorithm. We can conclude that globally (i.e. IL + Time) Micro-SOM is better than μ -Approx especially for large data sets.

In the near future we plan to explore the following lines:

- Use a hierarchical SOM to refine the partition obtained in STEP 1.
- Study the influence of different map topologies on the quality of the microaggregated data file.

References

1. Brand, R.: Microdata protection through noise addition. In: Inference Control in Statistical Databases, From Theory to Practice, London, UK, pp. 97–116. Springer, Heidelberg (2002)
2. Moore Jr., R.: Controlled data-swapping techniques for masking public use microdata sets. Technical report, Statistical Research Division Report Series, RR 96-04, US Bureau of the Census, Washington D.C. (1996)
3. Burridge, J.: Information preserving statistical obfuscation. Statistics and Computing 13(4), 321–327 (2003)
4. Domingo-Ferrer, J., Seb  , F., Solanas, A.: A polynomial-time approximation to optimal multivariate microaggregation. Comput. Math. Appl. 55(4), 714–732 (2008)
5. Adam, N.R., Worthmann, J.C.: Security-control methods for statistical databases: a comparative study. ACM Comput. Surv. 21(4), 515–556 (1989)
6. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
7. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge Based Systems 10(5), 557–570 (2002)
8. Oganian, A., Domingo-Ferrer, J.: On the complexity of optimal microaggregation for statistical disclosure control. Statistical Journal of the United Nations Economic Comission for Europe 18(4), 345–354 (2001)
9. Laszlo, M., Mukherjee, S.: Minimum spanning tree partitioning algorithm for microaggregation. IEEE Transactions on Knowledge and Data Engineering 17(7), 902–911 (2005)
10. Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogenous k -anonymity through microaggregation. Data Mining and Knowledge Discovery 11(2), 195–212 (2005)

11. Solanas, A., Martínez-Ballesté, A.: V-MDAV: Variable group size multivariate microaggregation. In: COMPSTAT 2006, Rome, pp. 917–925 (2006)
12. Solanas, A., Martinez-Balleste, A., Mateo-Sanz, J.M., Domingo-Ferrer, J.: Multivariate microaggregation based on genetic algorithms. In: 3rd International IEEE Conference on Intelligent Systems, pp. 65–70 (2006)
13. Martinez-Balleste, A., Solanas, A., Domingo-Ferrer, J., Mateo-Sanz, J.: A genetic approach to multivariate microaggregation for database privacy. In: IEEE 23rd International Conference on Data Engineering Workshop, April 17-20, pp. 180–185 (2007)
14. Solanas, A., Pietro, R.: A linear-time multivariate micro-aggregation for privacy protection in uniform very large data sets. In: Torra, V., Narukawa, Y. (eds.) MDAI 2008. LNCS (LNAI), vol. 5285, pp. 203–214. Springer, Heidelberg (2008)
15. Kohonen, T.: The self-organizing map. *Proceedings of the IEEE* 78(9), 1464–1480 (1990)
16. Kaski, S., Lagus, K.: Comparing self-organizing maps. In: Vorbrüggen, J.C., von Seelen, W., Sendhoff, B. (eds.) ICANN 1996. LNCS, vol. 1112, pp. 809–814. Springer, Heidelberg (1996)
17. Davies, D., Bouldin, D.: A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2(1), 224–227 (1979)
18. Kangas, J.: Increasing the error tolerance in transmission of vector quantized images by self-organizing map. In: FogelmanSoulie, F., Gallinari, P. (eds.) Proc. ICANN 1995, Int. Conf. on Artificial Neural Networks, pp. 287–291 (1995)
19. Vesanto, J.: SOM-Based data visualization methods. In: Intelligent Data Analysis, vol. 3, pp. 111–126 (1999)
20. Koikkalainen, P., Oja, E.: Self-organizing hierarchical feature maps. In: International Joint Conference on Neural Networks (IJCNN 1990), pp. 279–284 (1990)
21. Kangas, J., Kohonen, T., Laaksonen, J.: Variants of the Self-organizing map. *IEEE Transactions on Neural Networks* 1(1), 93–99 (1990)
22. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: Self-organizing map in Matlab: the SOM toolbox. In: Matlab DSP Conference, pp. 35–40 (1999)
23. Roussinov, D.G., Chen, H.: A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation. *Communication Cognition and Artificial Intelligence* 15, 81–112 (Spring 1998)
24. Vesanto, J.: Neural network tool for data mining: Som toolbox. In: Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET 2000), Oulu, Finland, Oulun yliopistonpiano, pp. 184–196 (2000)
25. Brand, R., Domingo-Ferrer, J., Mateo-Sanz, J.: Reference data sets to test and compare SDC methods for protection of numerical microdata. European Project IST-2000-25069 CASC (2002)

Identifying Clusters Using Growing Neural Gas: First Results

Riccardo Rizzo and Alfonso Urso

ICAR-CNR
Italian National Research Council,
Viale delle Scienze,
90128 Palermo, Italy
{ricrizzo,urso}@pa.icar.cnr.it
<http://www.pa.icar.cnr.it>

Abstract. Growing Neural Gas is a self organizing network capable to build a lattice of neural unit that grows in the input pattern manifold. The structure of the obtained network often is not a planar graph and can be not suitable for visualization; cluster identification is possible only if a set of not connected subgraphs are produced. In this work we propose a method to select the neural units in order to extract the information on the pattern clusters, even if the obtained network graph is connected. The proposed method creates a new structure called Labeling Network (LNet) that repeats the topology of the GNG network and a set of weights to the links of the neuron graph. These weights are trained using an anti-Hebbian algorithm obtaining a new structure capable to label input patterns according to their cluster.

Keywords: Growing Neural Gas, Self organization, Cluster Identification.

1 Introduction

What we call self organization is a mechanism that changes a complex interconnected structure of simple units in something completely different through the emergence of a global order out of local simple interactions. Often self organization is related to a different scale of observation: at the lower level the only observable thing is the connection of simple units, but if the point of view changes a completely new behavior can be observed.

An example is the Self Organizing Map (SOM) [1] where, at the lower level the units try to approximate the input patterns, and, on the upper level, we see the patterns organized in a topographic map.

Many other neural networks with similar mechanisms were developed after the SOM: Neural Gas (NG) [2] and Growing Neural Gas (GNG) [3] are among them. Growing Neural Gas is an interesting algorithm that is able to "produces" new neurons when it is necessary to represent new input patterns. Another characteristic of this algorithm is that neurons are connected using links that

are created during training, when the neurons are excited together. The training stage of the GNG network produce a network of units that completely fills the input pattern manifold. This can be clearly observed in the original paper [3] and will be clear in the following section. A complex and not planar neural graph, as the one produced by GNG, is difficult to use in clustering applications because there is not visualization (the network of the neural units has a complex structure even for simple datasets) nor cluster identification (see Fig. 2 for example).

In the past few approaches to cluster analysis using GNG were proposed. One method obtains a not connected graph using an edge removal procedure based on an utility factor and a suitable threshold [8]. Another method was focused on the GNG algorithm robustness and outliers filtering. The Minimum Description Length principle was then used to obtain the number of clusters on several attempts with different number of neural units [9].

In this work we propose a neural structure, called Labeling Network (LNet), that has the same topology of a trained GNG but uses an algorithm inspired by the EXIN mechanism (Excitatory-INhibitory) [6] in order to have one or few "active" neurons for each pattern cluster.

The paper is organized as follows: in the next section the details of the problem are explained with reference to a particular dataset and the GNG algorithm is introduced, then the proposed method is explained and in section 4 the algorithm is presented. The last two sections present some results and conclusions.

2 Self Organizing Networks and Cluster Identification

The goal of algorithms like SOM, NG and GNG is to approximate the input patterns at the best, in order to minimize the approximation error (this error is in [2] for NG and can be found as energy function in [7] for a modified SOM network). In all these algorithms communication among neurons is a fundamental part of learning algorithm: in SOM this exchange is supported by the neighborhood function defined over the lattice, in GNG networks the neural units exchange information about weight update using links, in NG the weight modification is propagated (with attenuation) from the best matching unit (*bmu*) to the neighborhood neurons. This communication among neurons is the basic mechanism of self organization.

From now on we focus on GNG network. The GNG network starts from a set of two neural units $U = \{u_a, u_b\}$ each of them with a set of weights $\mathbf{w}_i \in \Re^n$ and a cumulative error variable $e_i \in \Re$. These neurons are connected using a link $c_{a,b}$, that has an age $a_{a,b} \in \Re$ but not a weight. The training algorithm starts picking a random initial pattern $\mathbf{x} \in X \subset \Re^n$ and finding the best matching unit (*bmu*) u_{bmu} and the second *bmu*, u_{bmu2} , from the neuron set:

$$bmu = \arg \min_{u_i \in U} d(\mathbf{x}, \mathbf{w}_i), \quad bmu2 = \arg \min_{u_i \in U, u_i \neq u_{bmu}} d(\mathbf{x}, \mathbf{w}_i). \quad (1)$$

The edge $c_{bmu,bmu2}$, that connects u_{bmu} and u_{bmu2} , is set to zero or is created, then the error of *bmu* is updated as $\Delta e_{bmu} = \|\mathbf{w}_{bmu} - \mathbf{x}\|^2$ together with its

weight $\Delta\mathbf{w}_{bmu} = \varepsilon_b(\mathbf{x} - \mathbf{w}_{bmu})$, where ε_b is the learning parameter. The weight correction is propagated to the units connected as $\Delta\mathbf{w}_n = \varepsilon_n(\mathbf{x} - \mathbf{w}_n)$. At the end of the procedure all the connections emanating from bmu are updated adding a constant value, and all the error variables are decreased by multiplying for a constant $d < 1$. The edges that has an age larger than age_{MAX} are removed together with the units that have no emanating links.

Each λ learning steps a new unit u_r is added between the unit with the maximum accumulated error u_q and its neighborhood with the larger error variable u_f ; the weights \mathbf{w}_r of the new unit are mediated between \mathbf{w}_q and \mathbf{w}_f and the error of units u_q and u_f are decreased multiplying for a constant $\delta < 1$; finally $e_r = e_q$.

When the learning stage ends each neural unit approximate a subset of the input pattern; the number of patterns associated to a single neuron depends on the number of available neurons and it is roughly the same for each neuron. The neural units in some way represent the input patterns that are inside the corresponding Voronoi regions. Now we want to reduce the number of neurons that activate when the input patterns are presented, in order to identify the clusters inside the patterns set and produce few clear signals that allows to recognize the input patterns. This problem can be easily visualized using a real dataset.

The “Iris” dataset [10] is maybe the simplest and most common real dataset available. It is made of three clusters of objects in four dimensions: an idea of the spread of the pattern in their space is given in Fig. 11. This dataset was chosen because the two clusters “Iris Versicolor” and “Iris Virginica” are somewhat difficult to separate.

Training GNG on this dataset (for example with the following parameters $\epsilon_b = 0.5$, $\epsilon_n = 0.2$, $age_{MAX} = 50$, $\lambda = 50$, $d = 0.8$, $\delta = 0.5$ and 30000 learning steps; the meaning of the parameters can be found in [3]) produce the typical result in Fig. 2: a set of connected neurons (50 in the present trial). The two clusters “Iris Virginica” and “Iris Versicolor” are connected even if a separation among them can be easily spotted. This situation is no more than a simplified snapshot of the dataset (see Fig. 2).

3 Overview of the LNet Structure

In [4] the hidden units of a backpropagation network are considered as candidate hypothesis and trained using backpropagation algorithm, then the training is refined in order to better represent the target concept. The same consideration can be made for a trained GNG network: all the neurons are candidates to be the cluster centers for the clusters in input pattern set, what we need is a mechanism to select these neurons. To have emerging cluster centers from a set of neurons it is necessary that the activation of one neuron rise, and inhibits the activation of the other neurons.

The proposed Labeling Network has the same topology of the GNG, as depicted in right side of Fig. 3: neurons of the LNet receive an input E_i (given by

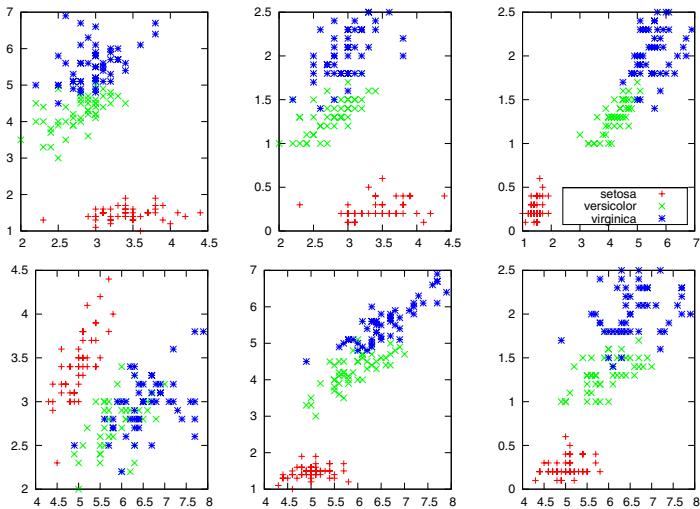


Fig. 1. The iris dataset plotted using all the combination of the four dimensions. It is possible to see that it is difficult to separate “Iris Versicolor” from “Iris Virginica”, while “Iris Setosa” forms a well separated cluster.

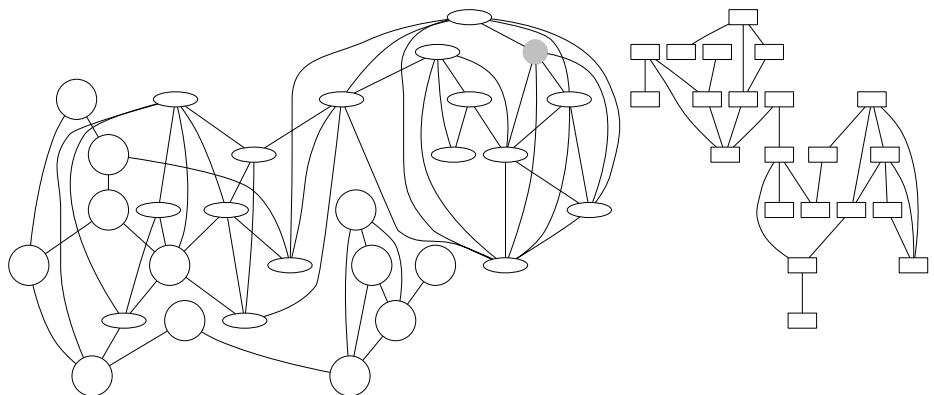


Fig. 2. The result of the learning of the GNG using the iris dataset. The units represented as boxes classify the “Iris Setosa”, the “circle” units classify the “Iris Versicolor” and the “ellipse” units the “Iris Virginica”; the gray unit is the “dead unit”, the one that does not classify any pattern.

eq. 2), from the trained GNG in the lower layer; these neurons have an activation a_i and excites and inhibits each other using the available links that are modulated by a new set of weights p_{ij} , see left side of Fig. 3. These weights are trained using an anti-Hebbian rule, and the activation is calculated using an iterative procedure that is explained in the next section.

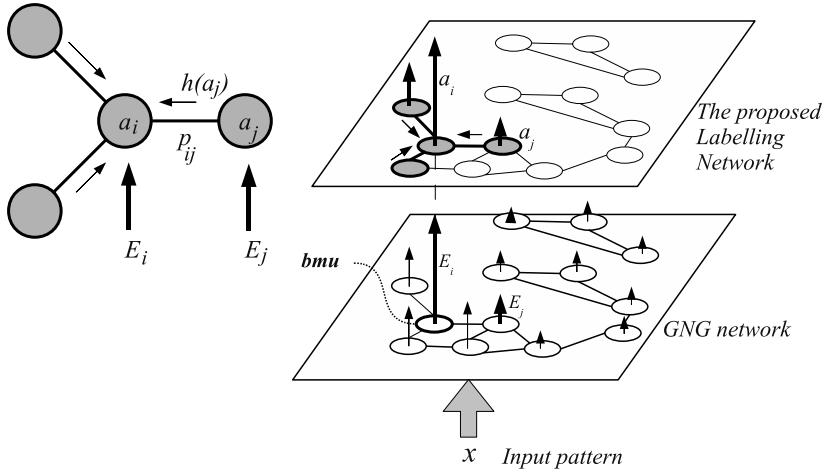


Fig. 3. A representation of the GNG and the Labeling Network on the right and a magnification of the interaction between two neurons, i and j , in the LNet network on the left

In order to build a cluster identification system it is necessary to use a two stage procedure that will produce the two layers on the right of Fig. 3:

1. train a GNG network using the available pattern set;
2. add the new LNet structure with the same topology of the GNG.

In the following we will refer only to this second structure, assuming that we have the trained GNG.

At the end of the training procedure we should submit an input to the network and obtain the activation of a single unit that will label the input pattern. In iris dataset, that is made of 3 clusters, 50 pattern each (150 pattern), we should ideally have 3 neural units that have the maximum activation, and "label" the input patterns. This is the kind of result we show in section 5.

4 The Proposed LNet Algorithm

In our network the activation a_i of the single neuron i is not related only to the similarity between the input pattern and the weight vector of the neuron, but it is calculated using a shunting equation that takes into account the excitation due to the input stimulus E_i and the excitation that came from the activation of the other connected neurons I_i using the weights p_{ij} as indicated in the left side of Fig. 3.

The excitation from the external input E_i in LNet neurons is calculated using the distance of the input pattern x from the weight vector of the neuron in GNG network w_i , according to:

$$E_i = \alpha \frac{d_{MAX} - d(\mathbf{w}_i, \mathbf{x})}{d_{MAX} - d_{min}}. \quad (2)$$

where $\alpha < 1$ is a coefficient necessary in order to do not saturate the neuron input, $d(\mathbf{w}_i, \mathbf{x})$ is the Euclidean distance between the pattern \mathbf{x} and the weight vector \mathbf{w}_i of the neuron i , and d_{MAX} and d_{min} are the maximum and minimum distances $d(\mathbf{w}_j, \mathbf{x})$ among the neurons j of the GNG network. This is necessary because we want the maximum excitation for the nearest neuron, usually called *bmu* (best matching unit).

The excitation due to other neurons is obtained from:

$$I_i = \sum_{j=1}^n h(a_j(t)) * p_{ij}. \quad (3)$$

where $h(a_i) = \max(a_i, 0)$ and p_{ij} is the weight of the connections.

As said before the input components E_i and I_i are used to calculate the activation of the single neuron using a shunt equation:

$$\frac{da_i}{dt} = -Aa_i(t) + \beta(1 + a_i(t))E_i - \gamma(C + a_i(t))I_i. \quad (4)$$

where A is the decay factor, β is the parameter that regulates the activation due to the external inputs, i.e. the term $(1 + a_i(t))E_i$ that also fixes the maximum value of $a_i(t)$ to 1, and γ is the parameter that regulate the inhibition from other neurons to the neuron activation, $-C$ is the minimum value of $a_i(t)$. The weights p_j are modified using an anti-Hebbian rule:

$$\frac{dp_{ij}}{dt} = \eta h(a_j(t)) * [h(a_i(t)) - p_{ij}]. \quad (5)$$

where η is the learning rate.

The equations 4 and 5 are discretized in order to carry on the simulations 5, and the time quantum Δt is added as a parameter of the simulation:

$$\Delta a_i(t+1) = -A * a_i(t) * \Delta t + \beta(1 + a_i(t))E_i * \Delta t - \gamma(C + a_i(t))I_i * \Delta t. \quad (6)$$

$$\Delta p_{ij}(t+1) = \eta h(a_j(t)) * [h(a_i(t)) - p_{ij}]. \quad (7)$$

the new values $a_i(t+1)$ and $p_{ij}(t+1)$ are calculated as: $a_i(t+1) = a_i(t) + \Delta a_i(t+1)$ and $p_{ij}(t+1) = p_{ij}(t) + \Delta p_{ij}(t+1)$. The learning algorithm of LNet is below. Note that the activation of the units is calculated using an interactive procedure that stops when the update of the solution is below the threshold ϵ .

The learning algorithm

1. $t = 0$
2. Select the input pattern x
3. Calculate the excitation E_i of the neurons
4. do
 - Calculate the excitation I_i of the neurons
 - Calculate $\Delta a_i(t)$ using eq.6 and the new $a_i(t+1)$ value
- while($\sum_i \Delta a_i(t+1) > \epsilon$)

5. update the weights p_{ij} using eq. 7
6. $t++$
7. if ($t < t_{max}$) goto 2 else exit

After the training the network is ready to classify the input patterns, for each pattern in input the activation of the neurons is calculated using the steps 3 and 4 of the learning algorithm.

5 Results

We show a typical result obtained with “Iris dataset”. As is possible to see from the foreground black bars in the graphs in Fig. 4 all the units of the GNG network are *bmu* for a numbers of patterns that varies from 0 to 10 (there is only one dead unit). In LNet the activation of the neurons is calculated using eq. 4 (step 3 and 4 of the algorithm) and this completely changes the landscape: as it is possible to see from the gray bars in the graphs, for all the patterns very few neurons are always activated, these neurons can clearly identify the input clusters. The results are repeated in the table down right of Fig. 4.

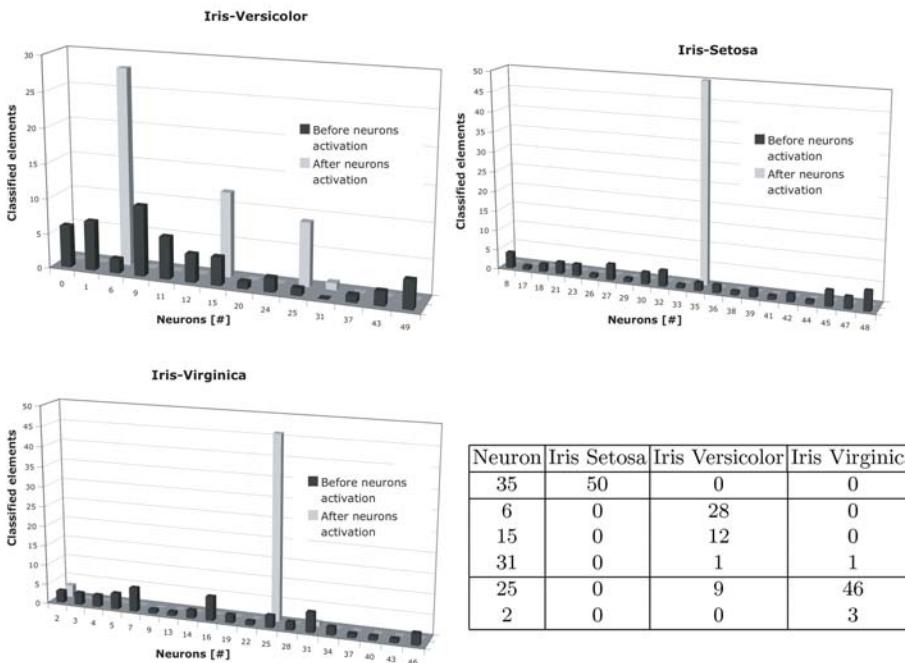


Fig. 4. Units of the GNG network representing the candidates to be cluster centers (before units activation) and emerging exemplars for the Iris pattern set

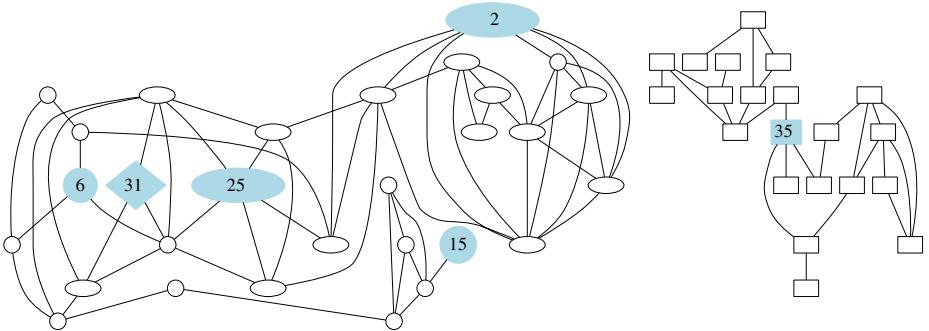


Fig. 5. The neural units that are activated with the input patterns. The gray units are the active one reported in the table down right of Fig. 4. The circle units (6 and 15) are the ones that label the “Iris Versicolor” cluster; the diamond unit 31 is the one that label one pattern “Iris Versicolor” and one pattern “Iris Virginica”; the oval units (25 and 2) are the one that label “Iris Virginica” patterns; the square unit 35 is the one that label the “Iris Setosa” cluster.

It is possible to notice that for the cluster Iris-Setosa there is only one exemplar that emerges, the unit number 35. For “Iris Versicolor” and Iris Virginica there is a small number of emerging exemplars, some of them are mixed (unit 25 and 31), although for “Iris Versicolor” most of the elements are classified by unit 6 and for Iris Virginica most of elements are classified by unit 25. If we assume that the unit 25 label “Iris Virginica” then we will have 9 patterns of “Iris Versicolor” that will be misclassified (the one in the bottom graph in Fig. 4).

In our experiments, to avoid the initial saturation of the neuron input, we carried out various trials varying values of the α coefficient defined in eq. 2, the best results were obtained for $\alpha = 0.3$. The other parameters were setup as follows: $A = 10.0$, $C = 10$, $\gamma = 1.0$, $\beta = 1.0$, the threshold $\epsilon = 0.00005$, $\eta = 0.6$ and $\Delta t = 0.00014$.

Some final remarks can be made looking at the topological position of the activated units that is reported in fig. 5 for cluster of “Iris-Setosa” the activated unit is in a “center” position: if we remove the units and the connected edges the subgraph is split in two parts; “Iris Virginica” are classified by the ellipse shaped units and the number 25 (that makes the highest bar in the histogram) seems to be at the crossing with the “Iris Versicolor” cluster; the round units are the ones that cluster “Iris Versicolor”. One of them (number 6) is again at the crossing with the cluster of “Iris Versicolor” while the other (number 15) is at the border of the cluster. The proposed algorithm seems to privilege the units that have a small number of active neighborhoods, This needs further investigations.

Another observation come from trials on other datasets that are not compact as the “Iris” dataset. For example in the “2 Rings” artificial dataset in Fig. 6(a) GNG is capable to grow a graph with two connected parts as the one in Fig. 6(b) but the LNet is unable to isolate two neurons that can label the patterns in the two clusters, the neurons that are activated are the same of the GNG network.

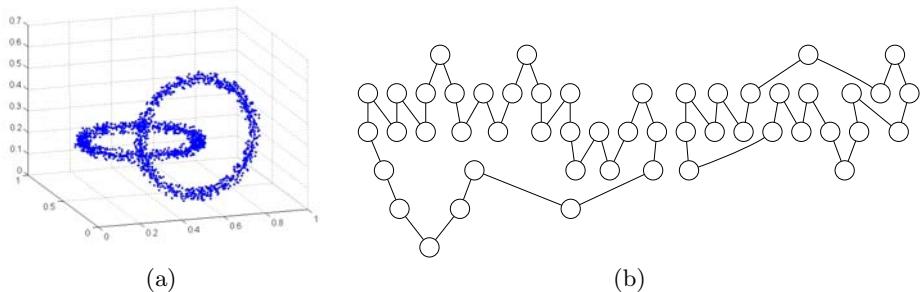


Fig. 6. 6(a) The “2 Rings” dataset and the GNG structure obtained 6(b)

6 Conclusions and Future Works

In this paper a new neural network capable to identify clusters from trained GNG is proposed. The new network have the same topology of the GNG but works in a very different way because has weight on the connections between neurons and take the input from excitation of the neurons in GNG. The network is capable to identify the clusters by selecting and activating few neurons that “label” the input patterns and allows to clearly identify clusters. The network was tested on the “Iris dataset” with good results, but more investigations are needed in order to understand if there is some meaning in the position of the active neural units. Moreover while the algorithm is able to isolate an active neuron for a globular cluster it is still difficult to identify clusters in some artificial datasets.

References

1. Kohonen, T.: Self Organizing Maps. Springer, Heidelberg (1997)
2. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: Neural Gas Network for Vector Quantization and its Application to Time-Series Prediction. *IEEE Trans. on Neural Networks* 4(4), 558–569 (1993)
3. Fritzke, B.: A growing neural gas network learns topologies. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) *Advances in Neural Information Processing Systems*, pp. 625–632. MIT Press, Cambridge (1995)
4. Fu, L.: A Concept Learning Network Based on Correlation and Backpropagation. *IEEE Trans. on System, Man and Cybernetics part. B* 29(6), 912–916 (1999)
5. Barreto, G.A., Arajo, A.F.R.: The role of excitatory and inhibitory learning in EXIN networks. In: *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 2378–2383. IEEE Press, Los Alamitos (1998)
6. Marshall, J.A.: Adaptive perceptual pattern recognition by self-organizing neural networks: Context, uncertainty, multiplicity and scale. *Neural Networks* 8, 335–362 (1995)
7. Heskes, T.: Energy functions for self-organizing maps. In: Oja, E., Kaski, S. (eds.) *Kohonen Maps*, pp. 303–316. Elsevier, Amsterdam (1999)

8. Costa, J.A.F., Oliveira, R.S.: Cluster Analysis using Growing Neural Gas and Graph Partitioning. In: Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA (2007)
9. Qin, A.K., Suganthan, P.N.: Robust growing neural gas algorithm with application in cluster analysis. Neural networks 17(8-9), 1135–1148 (2004)
10. Iris Dataset in Orange website,
<http://www.ailab.si/orange/doc/datasets/iris.htm>

Hierarchical Architecture with Modular Network SOM and Modular Reinforcement Learning

Masumi Ishikawa and Kosuke Ueno

Department of Brain Science and Engineering,
Kyushu Institute of Technology,
Kitakyushu 808-0196, Japan
ishikawa@brain.kyutech.ac.jp
<http://www.brain.kyutech.ac.jp/~ishikawa>

Abstract. We propose a hierarchical architecture composed of a modular network SOM (mnSOM) layer and a modular reinforcement learning (mRL) layer. The mnSOM layer models characteristics of a target system, and the mRL layer provides control signals to the target system. Given a set of inputs and outputs from the target system, a winner module which minimizes the mean square output error is determined in the mnSOM layer. The corresponding module in the mRL layer is trained by reinforcement learning to maximize accumulated future rewards. An essential point, here, is that neighborhood learning is adopted at both layers, which guarantees a topology preserving map based on similarity between modules. Its application to a pursuit-evasion game demonstrates usefulness of interpolated modules in providing appropriate control signals. A modular approach to both modeling and control proposed in the paper provides a promising framework for wide-ranging tasks.

Keywords: Modular network SOM, modular reinforcement learning, hierarchical architecture, pursuit-evasion game.

1 Introduction

Self organizing maps (SOM) are popular for their effectiveness in data visualization and topology preservation. A modular network SOM (mnSOM) is its extension in that it uses a function module as its element instead of a vector unit in SOM to increase its learning and representation capability [1][2]. Owing to topographic mapping of function modules on an mnSOM layer, neighboring function modules tend to have similar characteristics like SOM.

A contrasting approach is the conventional competitive learning; only a winner unit (or module) is highlighted, and similarity between units (or modules) is not taken into account. The present paper focuses on modules, not units.

There are two types of “interpolation.” The one is interpolation of outputs calculated from outputs of relevant modules, and the other is creation of an interpolated module from similar modules. Let the former be called “output interpolation,” and the latter be called “module interpolation” [3].

In the standard SOM and mnSOM, output interpolation is not used, and usually a winner module represents a prototype feature. Taking advantage of this characteristic, it was applied to segmentation in robotics [3][4]. In the standard SOM, module interpolation (actually unit interpolation) is rarely used, because the number of samples is usually much larger than that of units. In contrast to this, module interpolation is frequently used in the standard mnSOM, taking advantage of similarity between neighboring modules. Owing to a similarity measure between modules, SOM and mnSOM can naturally be extended to use output interpolation, when more precise representation is required.

On the other hand, in the standard competitive learning, neither output interpolation nor module interpolation is used. The modular selection and identification for control (MOSAIC) model uses the soft-max function, which realizes the output interpolation based on the responsibility signals [5]. Similarity between modules, however, is not explicitly represented. Furthermore, the number of modules in MOSAIC need to be the same with that of a target system for computational stability [6]. Therefore, it cannot use module interpolation.

Doya et al. proposed a modular reinforcement learning (RL) architecture composed of multiple modules for state prediction models and those for RL controllers [7]. Responsibility signals are used for output interpolation, and for learning of prediction models and RL controllers. Because similarity between modules is not explicitly defined, module interpolation is not used.

We propose a hierarchical architecture composed of an mnSOM layer and a modular reinforcement learning (mRL) layer. The mnSOM layer models characteristics of a target system, and the mRL layer provides control signals to the target system. Given a set of inputs and outputs of a target system, a winner module which minimizes the mean square output error (MSE) is determined in the mnSOM layer. The corresponding module in the mRL layer is trained by reinforcement learning to maximize accumulated future rewards.

An essential point, here, is that neighborhood learning is adopted at both layers, which guarantees a topology preserving map based on similarity between modules. An additional advantage, crucial from a practical viewpoint, is computational stability in the mnSOM layer in contrast to the conventional competitive learning due mainly to careful assignment of a learning rate to modules and data. These advantages are expected to provide superior performance in modeling and control of a dynamically changing target system by switching of modules at both layers.

A pursuit-evasion game is selected as a testbed to evaluate the effectiveness of the proposed architecture. Section 2 presents a hierarchical architecture composed of two layers: an mnSOM layer and an mRL layer. Section 3 describes simulation experiments. Section 4 concludes the paper.

2 Hierarchical Architecture

2.1 Modular Network SOM

We briefly present the following four processes of an mnSOM algorithm [1][2].

- Evaluative process: The following pairs of inputs and target outputs are provided to mnSOM, and the corresponding output errors are evaluated.

$$\{x_i(t), y_i(t); i = 1, \dots, M; t = 1, \dots, T_i\}. \quad (1)$$

where $x_i(t)$ is the input at time t in class i , $y_i(t)$ is the target output at t in class i , M is the number of classes, and T_i is the number of samples in class i .

- Competitive process: A module with the minimum mean square output error is determined as the winner.
- Cooperative process: The learning rate, $\Psi_i^{(k)}(t)$, for module k and class i is defined by the following normalized neighborhood function, $\phi(r, t)$, centered at the winner module.

$$\Psi_i^{(k)}(t) = \frac{\phi(r(k, v_i^*), t)}{\sum_{i'=1}^M \phi(r(k, v_{i'}^*), t)}. \quad (2)$$

$$\phi(r, t) = \exp \left[-\frac{r^2}{2\sigma^2(t)} \right]. \quad (3)$$

$$\sigma(t) = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) e^{-\frac{t}{\tau}}. \quad (4)$$

where $r(k, v_i^*)$ is the Euclidean distance between module k and the winner module v_i^* on an mnSOM plane, σ_{\min} is the minimum radius of neighborhood, σ_{\max} is the maximum radius of neighborhood, and τ is a time constant for neighborhood decay.

- Adaptive process: Connection weights of module k , $\mathbf{w}^{(k)}$, are modified by the following backpropagation algorithm,

$$\Delta \mathbf{w}^{(k)} = -\eta \sum_{i=1}^M \Psi_i^{(k)}(t) \frac{\partial E_i^{(k)}}{\partial \mathbf{w}^{(k)}}. \quad (5)$$

where $E_i^{(k)}$ is the output error of module k in class i .

The above four processes are repeated until the connection weights and mnSOM converge. An important issue in designing mnSOM architecture is the selection of appropriate function modules and a similarity measure between modules. Feedforward neural networks are used for mapping from input to output, and recurrent neural networks are used for dynamical systems as function modules. The similarity between modules is measured by the mean square output error, i.e., the mean square of the difference between outputs in one module and that in another module given the same inputs.

2.2 Modular Reinforcement Learning

Q-learning is used here, because it skillfully formulates states and actions, which are essential in a pursuit-evasion game. It maximizes accumulated future rewards

by trial and error under delayed and intermittent external rewards from the environment. An episode terminates when a predator captures its prey or when the maximum number of steps is reached.

Q-learning is used for learning by both a predator and prey. Action-value function, $Q(s,a)$, represents accumulated future rewards starting from the state, s , and taking the action, a , and is updated as,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \quad (6)$$

where α is a learning rate, r_{t+1} is a reward from the environment at time $t+1$, and γ is a discount rate. A constant penalty is given to each action. The ϵ -greedy algorithm is adopted to promote exploration.

Neighborhood learning is applied not only at the mnSOM layer but also at the mRL layer. Both layers have the same arrangement of modules, and modules at the mnSOM layer have one to one correspondence with those at the mRL layer: the winner module at the mnSOM layer has the same location with the corresponding one at the mRL layer. A learning rate, α , in neighborhood learning is given by,

$$\alpha(r) = \alpha_0 \exp \left(-\frac{r^2}{2\sigma(t)^2} \right). \quad (7)$$

where α_0 is a learning rate for the winner module, $\sigma(t)$ is the radius of the neighborhood in Eq.(3), and r is the Euclidian distance between the winner module and the current module on an mRL plane.

3 Simulation Experiments

3.1 Pursuit-Evasion Game Task

To evaluate the effectiveness of the proposed architecture, a pursuit-evasion game in Fig. 1 is used as a testbed. In the task, the predator learns to capture the prey for its survival, and the prey learns to escape from the predator for its survival. The desire for existence is represented by an intrinsic effort to keep a battery of an agent from exhaustion. They also learn a map of the environment to make pursuit or evasion more efficient. The following is a prospective scenario of behaviors of agents.

When the level of its battery is sufficiently high, an agent efficiently learns the environment motivated by curiosity [8] and novelty [9]. Under this condition, curiosity, either specific or diversive, motivates the agent to learn the environment. When a novel object is observed, novelty motivates the agent to approach and closely observe it, which in turn motivates specific curiosity.

On the other hand, when the level of its battery becomes lower, the predator starts to pursue its prey for its survival by taking advantage of the knowledge on the environment acquired through curiosity and novelty [10].

The present paper focuses on pursuit and evasion, particularly on the selection of the best matching module in the mnSOM layer, and on learning of the corresponding module in the mRL layer for the best pursuit/evasion.

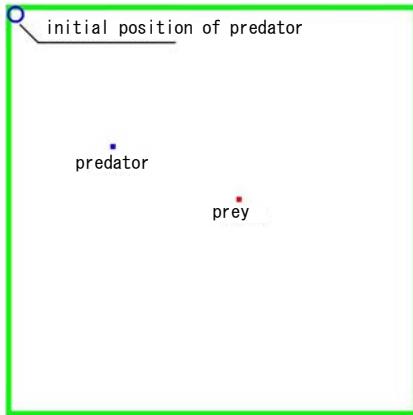


Fig. 1. The field for a pursuit-evasion game with 750×750 pixels surrounded by walls. The initial position of the prey is chosen randomly. The size of each agent is 10×10 pixels.

3.2 Learning at mnSOM Layer

Fig. 2 illustrates characteristic curves of the maximum speed of predators and prey. As time goes by, the speed of an agent gradually slows down and finally levels off at zero. Fig. 3 depicts characteristic curves of the distance between pairs of predators and prey. The zero distance indicates capture, and pursuit-evasion terminates. It is to be noted that characteristic curves are displaced vertically depending on the initial distance. In other words, the distance in Fig. 3 becomes larger or smaller depending on the initial distance. Fig. 4 portrays four characteristic curves used for mnSOM training, each corresponding to a combination of one predator and one prey. Vertical displacement also applies to these curves.

Table 1 gives values of parameters in mnSOM learning. The first five parameters are determined in a similar way with the standard SOM learning. The following parameter, i.e., the number of classes, is determined by the task in Fig. 4. Each characteristic curve in Fig. 4 is trained by backpropagation learning using a simple 3-layer neural network with five hidden units, given 30 samples from the corresponding curve.

Fig. 5 illustrates the resulting mnSOM after 5000 epochs. Four modules around four corners correspond to winner modules for four training data in Fig. 4. It well indicates that many interpolated modules are generated by the neighborhood learning, and that their characteristics change gradually from module to module. The color of each module indicates the average distance between adjacent modules, i.e., the darker the color, the longer the distance between adjacent modules. It is to be noted that there exists a large gap between the right part and the left part of the mnSOM plane; the right corresponds to slow prey (#1), and the left corresponds to fast prey (#4).

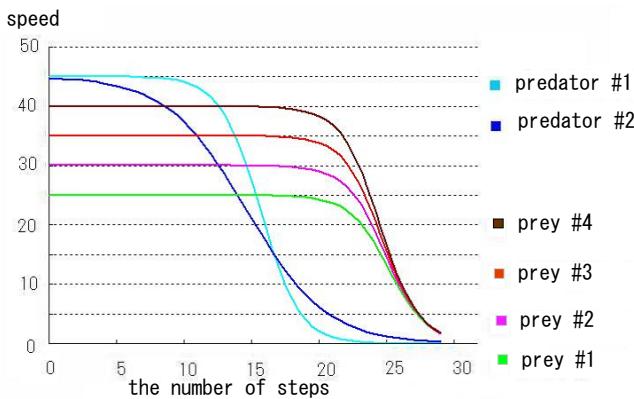


Fig. 2. Characteristic curves of the maximum speed of predators and prey over time. The predators run faster than the prey in the beginning, but slow down more quickly.

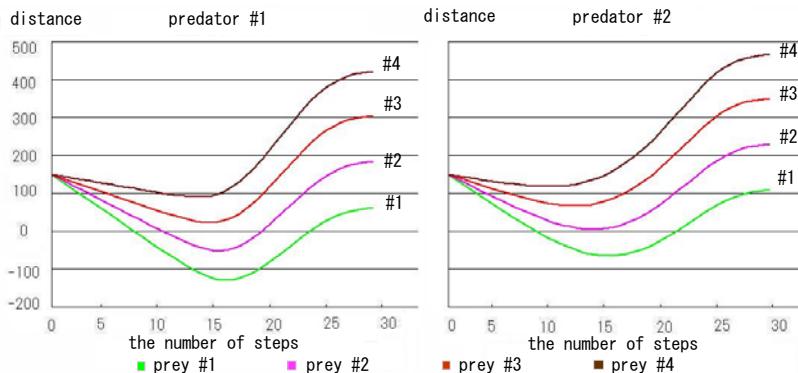


Fig. 3. Characteristic curves of the distance between the predator (#1 on the left, #2 on the right) and the prey (#1-#4)

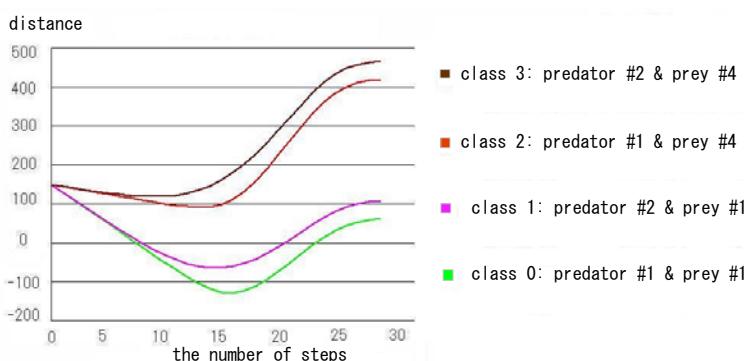


Fig. 4. Four characteristic curves obtained from two predators and two prey for mn-SOM training. Curves for classes 3, 2, 1, and 0 from the top to the bottom.

Table 1. Values of parameters in mnSOM learning and modular reinforcement learning

parameters	mnSOM	mRL
map size	10×10	10×10
learning rate	0.001	0.1
maximum epochs/episodes	5000	1000
initial radius of neighborhood	8	8
final radius of neighborhood	1.5	1
the number of classes	4	–
the number of states	–	30
the number of actions	–	2

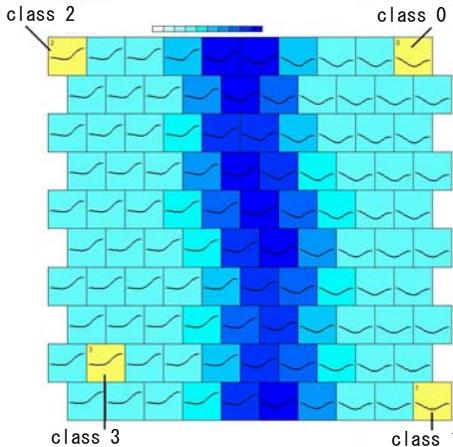


Fig. 5. The resulting mnSOM trained by classes 0, 1, 2, and 3 in Fig. 4. Each module contains illustration of its characteristic curve. The winner modules for training data are shown in yellow with the corresponding class number. The winner module for novel data, i.e., combination of predator #1 and prey #3, is located at the 5th module from the left in the top row.

3.3 Learning at Modular RL Layer

In Q-learning, how to define states and actions is important. Since there are two agents, i.e., a predator and prey, the number of states is at least the square of the number of grids in a robotic field, provided naive definition of states is used. This is computationally prohibitive, hence approximation aiming at decreasing the number of states is needed. In the present paper, state is defined by the quantized distance between the predator and the prey. Action of the predator is defined by either run at the maximum speed or slow move, and that of the prey is defined by either run at the maximum speed or standing still.

The reward for the predator is 5 at capture and -0.1 for each maximum speed run. The reward for the prey is 5 at escape and -0.1 for each maximum speed run. The ϵ -greedy policy with ϵ below is adopted to promote exploration.

$$\varepsilon = \frac{1}{\exp(an + b) - 1} \quad (8)$$

where n is an episode number, $a = 0.005$, and $b = 2.4$. Values of a and b are adjusted so as to reduce ε down to 10% of its initial value at around 450 episodes. An episode terminates when the predator captures the prey or the number of the maximum speed steps exceeds 30. Table 1 also gives values of parameters in modular reinforcement learning.

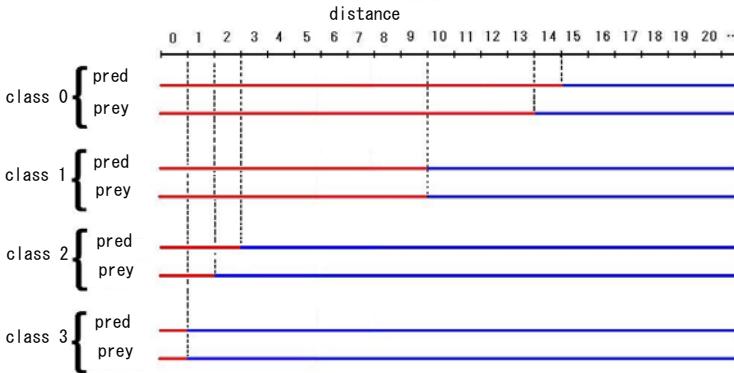


Fig. 6. The resulting optimal actions for 4 classes of training data. “pred” stands for “predator.” The red segment (on the left) stands for run at the maximum speed, and the blue segment (on the right) stands for slow move (predator) or standing still (prey).

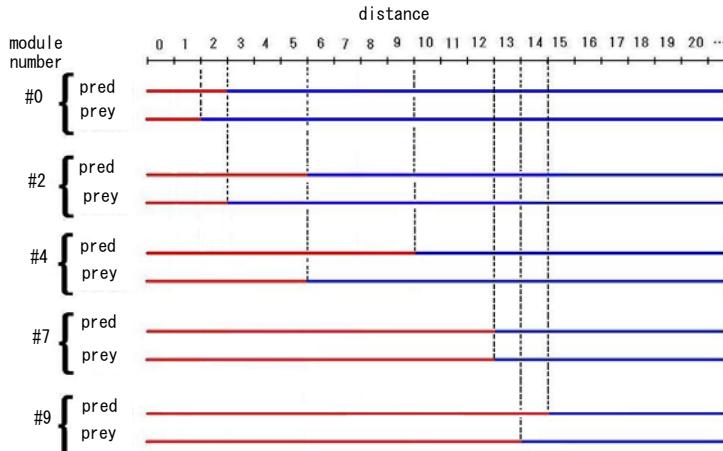


Fig. 7. The resulting optimal actions for interpolated modules trained by the neighborhood learning, i.e., without direct training data. #0 and #9 are trained modules, but are shown here for comparison. The red segment (on the left) stands for run at the maximum speed, and the blue segment (on the right) stands for slow move (predator) or standing still (prey).

Fig. 6 illustrates the resulting optimal actions for training data. Optimal action is determined by Q-values, i.e., action with larger Q-value is selected as optimal. Fig. 7 depicts the resulting optimal actions for interpolated modules trained by the neighborhood learning, i.e., without direct training data. Modules #0, #2, #4 and #7, #9 represent the 1st, 3rd, 5th, 8th, and 10th modules from the left, respectively, in the top row of Fig. 5. Module #0 and module #9 in Fig. 7 correspond to class 2 and class 0 in Fig. 5, respectively. Although only optimal actions are shown, Fig. 7 suggests that Q-values are topologically ordered on distance axis.

The winner module for novel data, i.e., combination of predator #1 and prey #3, is located at the 5th module from the left in the top row in Fig. 5. By using the interpolated module obtained by the neighborhood learning, 528 episodes out of 1000 are successful in capturing the prey. If this module is trained by directly giving data of predator #1 and prey #3, the number of successful catches increases up to 600 episodes. The fact that the success ratio is not so high is attributed to concurrent learning by both the predator and the prey. If only the predator learns, success of all episodes is confirmed.

In the standard technique, when novel data are given, the trained mnSOM module with the minimum MSE and the corresponding RL module are used as substitutes. Because the number of trained modules is much smaller than the total number of modules, control based on a substitute module is worse than that based on an interpolated module.

4 Conclusions and Discussions

We proposed a hierarchical architecture composed of a modular network SOM (mnSOM) layer and a modular reinforcement learning (mRL) layer. This hierarchical architecture provided precise prediction and effective control owing to modular structure and neighborhood learning at both layers. Its application to a pursuit-evasion game demonstrated usefulness of interpolated modules in providing appropriate control signals for novel data. This suggests that the proposed hierarchical architecture is able to provide appropriate control signals to a nonstationary target system by switching of modules.

Both mnSOM and mRL are computationally heavy, which is approximately proportional to the number of modules. However, the proposed hierarchical architecture is well worth the computational cost. When a task necessitates larger number of modules, multi-layer modules through abstraction for both mnSOM and mRL would become necessary.

Much remains to be done. More detailed and precise evaluation of interpolated modules is necessary such as the degree of linearity of interpolation in terms of Q-values and optimal actions. When information on the distance between a predator and prey is corrupted by noise, there can be two types of performance degradation. The former is misselection of the winner module at an mnSOM layer. Temporal filtering of observed distance reduces the noise, hence reduces the probability of misselection at the mnSOM layer. This also has direct bearing on

misselection of the corresponding module at an mRL layer. The latter is error in recognized state, i.e., quantized distance calculated from observed distance. This happens when distance error is larger than quantization error. Misrecognition of state degrades the performance of reinforcement learning.

Although a prospective scenario is “the predator starts to pursue its prey for its survival by taking advantage of the knowledge on the environment acquired through curiosity and novelty” as in [3,1], it has not yet been confirmed. Another issue is that although the proposed architecture can provide modeling and control of a given target system, sudden change of characteristics of the target system during an epoch has not been tried yet. The study on the detection of the change and the corresponding switching of modules is also left for future study.

In performance comparison in [3,3], only the comparison with the standard technique is given. However, [7] can use output interpolation, hence is expected to have superior performance to the standard technique. The performance comparison with [7] is left for future study. It is to be noted that the proposed architecture could also use output interpolation based on MSE.

Because two agents are involved in the current study, multi-agent reinforcement learning should be used instead of the conventional reinforcement learning. This inevitably increases computational cost, hence an efficient algorithm for approximate solution would be desired.

Acknowledgements

This research was partially supported by Grant-in-Aid for Scientific Research(C) (18500175) from the Ministry of Education, Culture, Sports, Science and Technology(MEXT), Japan.

References

1. Tokunaga, K., Furukawa, T., Yasui, S.: Modular network SOM: Extension of SOM to the realm of function space. In: Workshop on Self-Organizing Maps (WSOM), Kitakyushu, Japan, pp. 173–178 (2003)
2. Tokunaga, K., Furukawa, T.: Modular network SOM. Neural Networks 22, 82–90 (2009)
3. Muslim, M., Aziz, I.M., Furukawa, T.: Task segmentation in a mobile robot by mnSOM: a new approach to training expert modules. Neural Computing & Applications 6, 571–580 (2007)
4. Muslim, M.A., Ishikawa, M., Furukawa, T.: Task segmentation in a mobile robot by mnSOM and clustering with spatio-temporal contiguity. International Journal of Innovative Computing, Information & Control 5(4), 865–875 (2009)
5. Wolpert, D., Kawato, M.: Multiple paired forward and inverse models for motor control. Neural Networks 11, 1317–1329 (1998)
6. Minatohara, T., Furukawa, T.: The self-organizing adaptive controller for adaptability and generalization. IEICE Trans. J91-D(4), 1142–1149 (2008) (in Japanese)
7. Doya, K., Samejima, K., Katagiri, K., Kawato, M.: Multiple model-based reinforcement learning. Neural Computation 14, 1347–1369 (2002)

8. Schmidhuber, J.: Self-motivated development through rewards for predictor errors/improvements. In: Developmental Robotics, 2005 AAAI Spring Symposium (2005)
9. Markou, M., Singh, S.: Novelty detection: a review - part 1: statistical approaches. *Signal Processing* 83(12), 2481–2497 (2003)
10. Ishikawa, M., Hagiwara, T., Yamamoto, N., Kiriake, F.: Brain-inspired emergence of behaviors in mobile robots by reinforcement learning with internal rewards. In: HIS 2008, Barcelona, Spain, pp. 138–143 (2008)

Hybrid Systems for River Flood Forecasting Using MLP, SOM and Fuzzy Systems

Ivna Valen  a and Teresa Ludermir

Federal University of Pernambuco C.P 7851 - 50.732-970 - Recife - PE, Brazil
`{icbv,tbl}@cin.ufpe.br`

Abstract. This article presents an approach of data partitioning using specialist knowledge incorporated to intelligent solutions for river flow prediction. The main idea is to train the processes through a hybrid systems, neural networks and fuzzy, characterizing its physical process. As a case study, the results obtained with this models from three basins, *Tr  s Marias*, *Tucur  i* and *Foz do Areia*, all situated in Brazil, are investigated.

Keywords: Modular Models, Hybrid Systems, Self-Organizing Map, Multi-Layer Perceptron, Fuzzy Systems, River Flood Forecasting.

1 Introduction

Artificial Neural Networks (ANNs) are increasingly used in the water resources field and environmental sciences, especially for river flow prediction, a classic case of temporal series. The use of ANNs presents a satisfactory performance in many fields, such as: classification, associative memory, temporal series processing, etc. However, in general, these networks are being used as tools to make flow predictions without considering their physical particularities over time.

Forecasting hydrological variables, like river flow, water levels and rainfall is necessary in planning, design, maintenance and operation of water resources systems and consists of a complex process since it is a highly non-linear phenomenon. Companies that generate electricity execute a *Monthly Operational Program* that defines the generation of each unity of the company and the commercialization of energy interchange between them. An important factor for an optimized service for energy demand is the presence of an efficient inflow prediction system, since the future system capacity is influenced by future inflows that has an intrinsic stochastic (random) nature. So, the development of a methodology that improves such predictions is very important.

Real time inflow forecasting has applications in operational flood as well as drought management. It can forewarn extreme flood as well as drought conditions, and can help in optimum operation of reservoirs and power plants. The process of modeling the hydric behavior of a reservoir has fundamental importance for the planning and operation of a flow prediction system of hydric resources. Traditionally, the flow predictions have been made through deterministic models that attempt to describe the water movement behavior by the laws of physics,

resulting in a numeric system of differential equations. Alternatively, conceptual models have been used once they include equations that guide the drainage in a simplified way, but plausible enough to represent the desired behavior. However, these models in general require a big quantity of data, sophisticated optimization techniques for calibration and a detailed knowledge of the physical process. In the last years, many studies showed that neural networks presented themselves as efficient tools for flow prediction modeling [18][13][17][4][19][2][12][20][21]. The greatest advantage of using a neural network for flow prediction is its ability to learn the non-linear behavior of the transformation process of rainfall-runoff in a hydrographical basin by relating a set of entry variables (rain and/or past flows) and the output variables (generally, the flow itself) without the necessity of providing a mathematical equation or any consideration about the physical behavior of the hydrographical basin.

In a general way, the applications found in the literature try to represent the flow prediction process using a single neural network [5][12][15]. However, the flow presented in a hydrograph (flow as a function of time), exemplified in Fig. 1, is a product of different physical processes [7]. The hydrograph, in short, can be understood as a composition of an ascending branch that begins at point 1 (this point represents the end of the base flow, before the rain season). Between points 1 and 2, there are water infiltration and sub superficial flow, and at point 2, the superficial flow begins quickly increasing its value until moment 3 where it reaches its peak and then the descending branch of the hydrograph begins. The recession period of the hydrograph, between points 3 and 5, represents the gradual emptying of the rail river, but at points 3 and 4, there is a strong influence of the superficial flow still. Finally, from point 5, the flow returns to a general base flow from a superior point given the elevation of the water level.

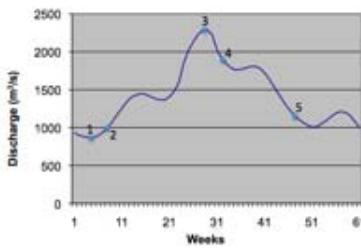


Fig. 1. Hydrograph (Discharge x Weeks)

Therefore, the use of a simple neural network may not be sufficient to represent this complex process of mapping between the inputs and outputs of a hidrogram. For that reason, the aim of this study is: implement two Modular Models that will be able to represent the physical process of ascent and descent of hidrogram for different events (high and low flows), then apply these methods to three basins located in the North, Northeast and South of Brazil, and finally, compare

the two proposed methodologies with the traditional methodology, i.e., a single neural network to represent the overall process.

2 Problem Formulation

The process of developing a composed model, that has local ones, where each one of them represents an answer for the system behavior, results in a series of operations [6] that are listed below:

- Selection of the events: must correspond to different aspects of the system behavior, for example, high and low flows, up and downs.
- Selection of models: the selected events can represent models of the same nature or different nature.
- Objective function definition: expresses the simulation quality.
- Model calibration: the model parameters must be trained to optimize the selected objective functions.
- Model combination: the local models are finally integrated in a composed model.

In this article, we go through the whole process to develop two composite models. The steps will be emphasized throughout the text.

3 Methodology

The methodology used in this work consists of a modular modeling scheme where the hydrograph is decomposed in four parts that represents the ups (ascension) and downs (recession), the high and low flows of the process (Fig. 2). This work proposes to use two modular models: SOM and MLP, and MLP with Fuzzy objective function. However, in literature, other ANNs are heavily studied and used to forecast time series [16], as the network TDNN (Time Delay Neural Networks), which is an ANN with time delay, and the Recurrent Networks [8].

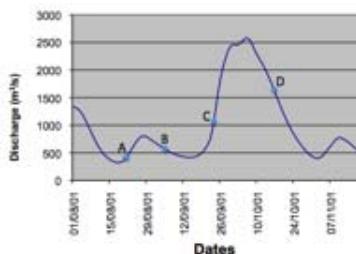


Fig. 2. Events of Hydrograph. A - Ascension low flow, B - Recession low flow, C - Ascension high flow, and D - Recession high flow.

For *Modular Model: SOM and MLP (MM_{SM})* we used two networks, self-organization map (SOM) and Multi-Layer Perceptron (MLP). The objective is to use the SOM network to characterize the various phenomena of the process and then use a MLP network for each of the groups defined by the SOM network.

For *Modular Model: MLP and Fuzzy objective function (MM_{MF})* we used Multi-Layer Perceptron network and Fuzzy logic in order to achieve the composition of the outputs of MLPs.

3.1 Self-Organizing Map (SOM)

The neural network self-organizing map (SOM) is classified as non-supervised and is characterized by the use of the input vector of each unit group as the prototype patterns associated with that unit [3]. During the process of self-organization, the unit vector whose weights most closely approximates to the input signal is chosen as winner. The winning unit and the units in their neighbourhood (according to the topology) have their weights updated.

The configuration of the SOM network is presented as follows: 14 inputs, where each entry corresponds to a daily flow and 4 neurons on the map, which determine the 4 groups of possible groups. Each entry is linked to all neurons.

3.2 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron (MLP) networks are composed of interconnected computable nodes arranged in layers that, in theory, can approximate any mathematical function, linear or nonlinear [3]. The basic architecture of an MLP used in this project is described by: an input layer, hidden layer and output layer.

3.3 Modular Model: SOM and MLP (MM_{SM})

As previously mentioned, the focus is on modularization based on hydrograph, instead of building a global model that is responsible for representing the flow of water in all schemes. Thus, the MM_{SM} proposed, would consist of 5 neural networks, which are: SOM, MLP₁, MLP₂, MLP₃ e MLP₄, where each of these networks has settings that are described in the topic concerning the experiments. First, the input data are inserted in the SOM, to be separated into 4 groups, where each of these groups will be included in an MLP, and finally, the output of MM_{SM} is the output of these MLPs (Fig. [3]).

3.4 Modular Model: MLP and Fuzzy Objective Function (MM_{MF})

In this model, the input data are divided into high and low flows. Each set is trained by a specific MLP, MLP_A will be called for high flows and MLP_B for low flows, where each of them has different configuration that will be shown in the experiment topic. After obtaining the results of the MLP_A and MLP_B, a concatenation is performed to obtain a single output using Fuzzy logic (Fig. [4]),

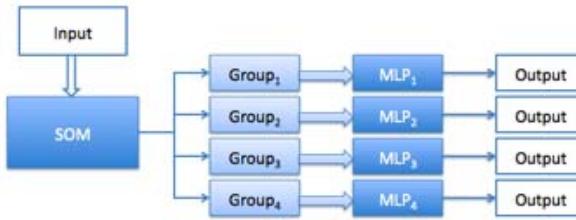


Fig. 3. Modular Model: SOM and MLP (MM_{SM})

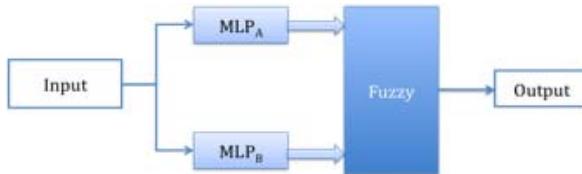


Fig. 4. Modular Model: MLP and Fuzzy (MM_{MF})

giving the impression of being an output of a global MLP. We use Fuzzy logic in this model to admit intermediate values between high and low flows.

The hidrogram for high or low flows may be in a process of ascension or recession. So, to take into account this fact we used in this work a fuzzification error for training each neural network (MLP_A and MLP_B). Then, for process of ascension the objective function was considered:

$$OF_{Ascension} = \sqrt{\frac{1}{n} \sum_{i=1}^n (Q_{c,i} - Q_{o,i})^2 \left(\frac{Q_{o,i} + \Delta Q}{Q_{o,max}} \right)^2} \quad (1)$$

Where: n = total number of examples; $Q_{o,i}$ = inflow observed in instant i ; $Q_{c,i}$ = calculated inflow in instant i ; $Q_{o,max}$ = maximum observed inflow; ΔQ = derivative of the hidrogram ($Q_{o,i} - Q_{o,i-1}$)

The hidrogram was considered in a position to rise (ascension) when the Rate of increase is greater than a given value α pre-set depending on the basin analyzed. For recession the objective function considered was:

$$OF_{Recession} = \sqrt{\frac{1}{n} \sum_{i=1}^n (Q_{c,i} - Q_{o,i})^2 \left(\frac{Q_{o,i} - \Delta Q}{Q_{o,max}} \right)^2} \quad (2)$$

In every moment of the training is calculated the Rate, as follow:

$$Rate = \frac{(Q_t - Q_{t-1})}{Q_{t-1}} 100 \quad (3)$$

Where: Q_t = current inflow; Q_{t-1} = previous inflow.

During the training an ascension is considered when the value of the Rate is more than α , which this article was considered 20%. On the other side a recession is considered when the Rate is less than 20%.

Thus, during the training of the neural network the error considered was one Fuzzy function, that uses $OF_{Ascension}$ and $OF_{Recession}$, according to the Rate. Therefore, when the Rate corresponds to a value x between the values α (20%) and $-\alpha$ (-20%) the error is fuzzified according to Fig. 5.

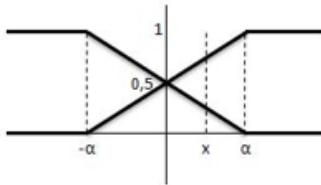


Fig. 5. Fuzzy Objective Function

In this case the composite objective function is:

$$OF_{Composite} = \frac{OF_{Ascension} Coef_{ascension} + OF_{recession} Coef_{recession}}{Coef_{ascension} + Coef_{recession}} \quad (4)$$

Where:

$$Coef_{ascension} = \frac{x + \alpha}{2\alpha} = \frac{x + 20}{40} \quad (5)$$

$$Coef_{recession} = \frac{x - \alpha}{-2\alpha} = \frac{x - 20}{-40} \quad (6)$$

4 Case Studies and Results

For the case study we used data (discharge) from three power plants: Tucuru  (North), Tr s Marias (Northeast), and Foz do Areia (South), located in Brazil. Some information about the barrages can be viewed in Fig. 6. The horizon weekly was used because it is used for planning the national energy sector. The series of daily flow data were divided into three data sets: first is to adjust the weights (50%), second set is for cross-validation (25%) and third set (25%) to assess the performance of the proposed methodologies. The historic of average daily flow used here corresponds to the period 1968 to 2004. The MLPs networks (MLP_1 , MLP_2 , MLP_3 , MLP_4 , MLP_A and MLP_B) for Tucuru , Foz do Areia and Tr s Marias had 14 input neurons (14 days previous), 5 neurons in hidden layer and 7 in the output layer, i.e., the forecasting horizon used was a week ahead. The learning-rate parameter (η) and momentum constant (α) were: $\eta = 0.30$ and $\alpha = 0.50$ for Tucuru , $\eta = 0.30$ and $\alpha = 0.40$ for Tr s Marias and $\eta = 0.40$ and $\alpha = 0.60$ for Foz do Areia.

	Basin name		
Information	Três Marias	Tucuruí	Foz do Areia
Draining area (km^2)	50,732	757,577	30,127
Normal minimum level (m)	549.20	51.64	700.00
Normal maximum level (m)	572.50	71.93	742.00
Normal maximum volume (hm^3)	19,528	45,500	5,779
Useful volume (hm^3)	15,278	32,013	3,804
Period	1969-2004	1969-2004	1969-2004
Localization	Northeast	North	South
Total power (MW)	396	8,340	1,676

Fig. 6. Hydrological Characteristics: Três Marias, Tucuruí and Foz do Areia

To find out how good are the results of the proposed models, we made a comparison between the results of a simple MLP (MLP_S), the proposed methodologies (MM_{SM} and MM_{MF}) of Modular Model and the results of the statistical methodology used by the electricity sector. To compare these methods we calculated the Mean Absolute Percentage Error (MAPE), given by:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \frac{(Q_{o,i} - Q_{c,i})}{Q_{o,i}} 100 \quad (7)$$

Methods		Três Marias	Tucuruí	Foz do Areia
MLP_S	Average	21,2%	8,7%	25,6%
	Standard deviation	18,0%	8,0%	9,0%
MM_{SM}	Average	20,8%	8,3%	19,1%
	Standard deviation	12,0%	7,5%	7,0%
MM_{MF}	Average	20,2%	8,3%	19,5%
	Standard deviation	11,0%	7,0%	6,5%
Statistician	Average	27,0%	11,5%	33,0%
	Standard deviation	19,0%	9,0%	15,0%

Fig. 7. Errors obtained in the reservoirs: Três Marias, Tucuruí and Foz do Areia

Looking at the table of errors found in Fig. 7, its possible to affirm that the results obtained with the proposed models are better than the other models. The graphs below (Fig. 8, 9, 10) that corroborates this assertion, show the comparasion results of the Modular Models (MM_{SM} and MM_{MF}) forecast, MLP_S forecast and of the actual outcome (observed).

In order to compare statistically the average results found by different models the test of hypothesis of difference between means was held. Thus, we conclude with 95% confidence (or a chance of error of 5%) that there is a difference between the average results of the statistical with MLP_S , and between MM_{SM} and MM_{MF} with MLP_S . Furthermore, we found no statistical differences between

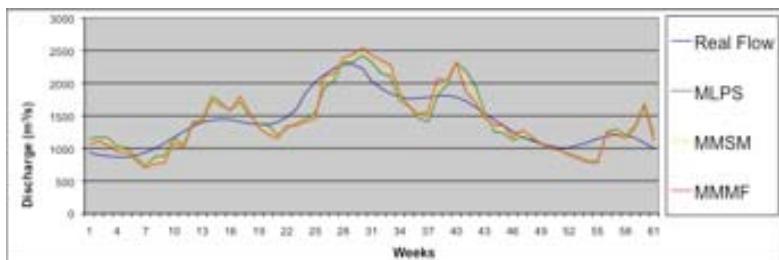


Fig. 8. Hydrograph of Tr s Marias: comparison of inflows

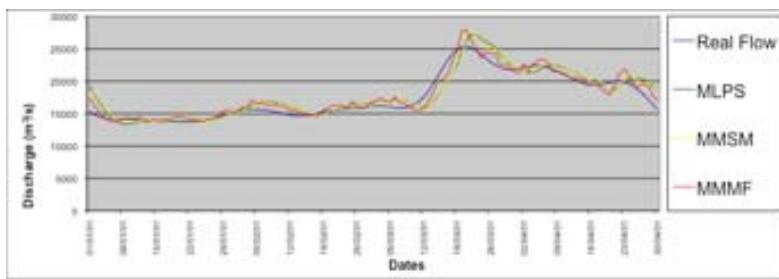


Fig. 9. Hydrograph of Tucurui: comparison of inflows

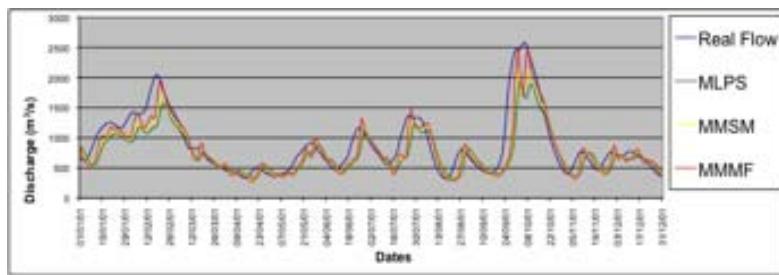


Fig. 10. Hydrograph of Foz do Areia: comparison of inflows

the Modular Models (MM_{SM} and MM_{MF}). This analysis was carried out for the three reservoirs studied and all leads to this conclusion. The time taken for the implementation of MM_{SM} was higher than the MM_{MF} , however, to obtain the results, both models showed similar performance.

5 Final Considerations

In this article, two approaches composed of neural networks and Fuzzy logic were presented: the Modular Models SOM and MLP, as well as the MLP and

Fuzzy, are used to represent the complex phenomenon of inflow forecast through the decomposition of its hydrograph. The genesis of the study presented here is based on the concept about which different segments of the complete hydrograph are a result of different physic processes that take place at the hydrographical basin and, therefore, need different modeling. The main conclusions of this study are:

1. The approach of decomposing the hydrograph in different segments can be a good alternative than trying to use a single neural network.
2. The results obtained through the use of different metrics shows that the proposed methodology has satisfactory results and that minimize problems of tendency (forecasts that are higher or lower than expected) in a particular portion of the flow mentioned by the authors [9][15][19].
3. Considering a fuzzy objective function allows a better representation of the physical phenomenon of ascension and recession of the hydrograph for high and low flows alike.
4. The obtained results for the three studied basins, Três Marias, Tucuruí and Foz do Areia, are satisfactory when compared with the use of a simple neural networks and very superior to statistical models used by the electric sector.

According to the achieved results, the following future works are proposed: consider a non-linear fuzzification; adopt four levels of fuzzification - soft up, abrupt up, soft down, abrupt down; and implement a Modular Model with a dynamic Neural Network, i.e., with memory.

Acknowledgments. The authors are thankful to Brazilian research councils FACEPE for financial support and Mêuser J. S. Valença for guiding us in the area of water resources.

References

1. Almeida, P.E.M., Evsukoff, A.G.: Intelligent Systems: Fundamentals and Applications (Sistemas Inteligentes: Fundamentos e Aplicações). cap. Fuzzy Systems, Manole, Barueru, São Paulo (2005)
2. Birikundavyi, S., Labib, R., Trung, H.T., Rousselle, J.: Performance of neural networks in daily streamflow Forecasting. *J. Hydrol. Engg.*, ASCE 7(5), 392–398 (2002)
3. Braga, A.P., Carvalho, A.C., Ludermir, T.B.: Artificial Neural Networks: Theory and Applications (Redes Neurais Artificiais: Teoria e Aplicações), 262 p. Rio de Janeiro, Livro Técnico e Científico (2000)
4. Campolo, M., Andreussi, P., Soldati, A.: River Flood Forecasting with Neural Network Model. *Wat. Resour. Res.* 35(4), 1191–1197 (1999)
5. Dawson, D.W., Wilby, R.: An artificial neural network approach to rainfall runoff modeling. *Hydrol. Sci. J.* 43(1), 47–65 (1998)
6. Fenicia, F., Savenije, H.H.G., Matgen, P., Pfister, L.: A comparison of alternative multiobjective calibration strategies for hydrological modeling. *Water Resources Research* 43(3) (2007)

7. Corzo, G., Solomatine, D.: Knowledge-based modularization and global optimization of artificial neural network models in hydrological forecasting. *Neural Networks* 20, 528–536 (2007)
8. Haykin, S.: *Neural Networks: A Comprehensive Foundation*, 2nd edn., p. 842. Prentice Hall, Englewood Cliffs (1998)
9. Hsu, K.-L., Gupta, H.V., Sorooshian, S.: Artificial Neural Network Modeling of the Rainfall-Runoff Process. *Wat. Resour. Res.* 31(10), 2517–2530 (1995)
10. Kartalopoulos, S.V.: *Understanding Neural Networks and Fuzzy Logic*. IEEE Press, Los Alamitos (1996)
11. Kosko, B.: *Neural Networks and Fuzzy Systems*. Prentice-Hall, Englewood Cliffs (1992)
12. Jain, A., Indurthy, S.K.V.P.: Comparative analysis of event based rainfall-runoff modeling techniques-deterministic, statistical, and artificial neural networks. *J. Hydrol. Engg.*, ASCE 8(2), 1–6 (2003)
13. Minns, A.W., Hall, M.J.: Artificial neural networks as rainfall runoff models. *Hydrol. Sci. Jour.* 41(3), 399–417 (1996)
14. Reyes, C.A.P.: Coevolutionary Fuzzy Modeling. In: Pe a Reyes, C.A. (ed.) *Coevolutionary Fuzzy Modeling*. LNCS, vol. 3204, pp. 51–69. Springer, Heidelberg (2004)
15. Sajikumar, N., Thandaveswara, B.S.: A non-linear rainfall-runoff model using an artificial neural network. *J. Hydrol.* 216, 32–55 (1999)
16. Sato, et al.: Learning chaotic dynamics by recurrent neural networks. In: Proceeding of the International Conference on Fuzzy Logic and Neural Nets, Iizuka, pp. 601–604 (1990)
17. Shamseldin, A.Y.: Application of a neural network technique to rainfall-runoff modeling. *J. Hydrol.* 199, 272–294 (1997)
18. Smith, J., Eli, R.N.: Neural Network Models of the Rainfall Runoff Process. *ASCE Jour. Wat. Res. Plng. Mgmt.* 121, 499–508 (1995)
19. Tokar, A.S., Markus, M.: Precipitation Runoff Modeling Using Artificial Neural Network and Conceptual models. *J. Hydrol. Engg.*, ASCE 5(2), 156–161 (2000)
20. Valen a, M.J.S.: Applying Neural Networks: a complete guide (*Aplicando Redes Neurais: um guia completo*), 264 p. Livro R pido, Olinda-PE (2005)
21. Valen a, M.J.S.: Fundamentals of Neural Networks: examples in Java (*Fundamentos das Redes Neurais: exemplos em Java*), 382 p. Livro R pido, Olinda-PE (2007)

Topographic Mapping of Astronomical Light Curves via a Physically Inspired Probabilistic Model

Nikolaos Gianniotis¹, Peter Tino², Steve Spreckley³,
and Somak Raychaudhury³

¹ Heidelberg Collaboratory for Image Processing,
University of Heidelberg D-69115 Heidelberg, Germany

² School of Computer Science,
The University of Birmingham,
Edgbaston B15 2TT United Kingdom

³ School of Physics and Astronomy,
The University of Birmingham,
Edgbaston B15 2TT United Kingdom

Abstract. We present a probabilistic generative approach for constructing topographic maps of light curves from eclipsing binary stars. The model defines a low-dimensional manifold of local noise models induced by a smooth non-linear mapping from a low-dimensional latent space into the space of probabilistic models of the observed light curves. The local noise models are physical models that describe how such light curves are generated. Due to the principled probabilistic nature of the model, a cost function arises naturally and the model parameters are fitted via MAP estimation using the Expectation-Maximisation algorithm. Once the model has been trained, each light curve may be projected to the latent space as the mean posterior probability over the local noise models. We demonstrate our approach on a dataset of artificially generated light curves and on a dataset comprised of light curves from real observations.

Keywords: Topographic mapping, eclipsing binary stars.

1 Introduction

The Generative Topographic Map algorithm (GTM) [1] has been introduced as a probabilistic analog to SOM [2], seeking to address certain of its limitations such as the absence of a cost function. The GTM formulates a mixture of spherical Gaussians densities constrained on a smooth image of a low-dimensional latent space. Each point in the latent space is mapped via a smooth non-linear mapping to its image in the high-dimensional data space. This image plays the role of the mean of a local spherical Gaussian noise model that is responsible for modelling the density of data points in its vicinity. The GTM can be readily extended to structured data by adopting alternative formulations of noise models in the place of Gaussian densities. Such extensions have been proposed in

[3] for the visualisation of symbolic sequences and in [4] for the visualisation of tree-structured data.

Here we present a further extension of the GTM to a novel data type, namely light curves that originate from eclipsing binary systems. Binary stars are gravitationally bound pairs of stars that orbit a common centre of mass. Astronomical observations suggest that almost half of the stars are binary ones. Thus, studying such systems procures knowledge for a significant proportion of stars. Binary stars are important to astrophysics because they allow calculation of fundamental quantities such as masses and radii, and are important for the verification of theoretical models for stellar formation and evolution. A particular subclass of binary stars are eclipsing binary stars. The luminosity of such stars varies over time and forms a graph called light curve. Light curves are important because they provide information on the characteristics of stars and help in the identification of their type.

2 Physical Model for Eclipsing Binaries

The physical model that generates light curves from eclipsing binary systems is described by the following set of parameters: mass $M_1 \in [0.5, 100]$ (in solar masses) of the primary star (star with highest mass of the pair), mass ratio $q \in [0, 1]$ (hence mass of secondary star is $M_2 = qM_1$), eccentricity $e \in [0, 1]$ of the orbit and period $\rho \in [0.5, 100]$ measured in days, all of which specify the shape of the orbit. Furthermore, two angles describing the orientation of the system are necessary [5] which are known as the inclination $i \in [0, \frac{\pi}{2}]$ and the argument of periastron $\omega \in [0, 2\pi]$ (see Fig. II). Inclination describes the angle between the plane of the sky and the orbital plane and periastron is the angle $\omega \in [0, 2\pi]$ that orients the major axis of the elliptic orbit within its plane, that is ω is measured within the orbital plane. Finally, a third angle known as the longitude of ascending node ($\Omega \in [0, 2\pi]$) is necessary for the complete description of a binary system. However, since it has no effect on the observed light curves, we omit it from the model. We collectively denote these parameters by vector θ .

The mass M of each star relates to the luminosity L radiated by a surface element [5] of the star according to $L = M^{3.5}$. Moreover, masses relate to the radii R of the stars via:

$$R = \begin{cases} 10^{0.053+0.977 \log_{10}(M)}, & \text{if } M < 1.728; \\ 10^{0.153+0.556 \log_{10}(M)}, & \text{otherwise.} \end{cases} \quad (1)$$

These relations show that the primary star is the most luminous one and the one with the greatest area of the pair (a star appears as a disc to an observer). Thus, the *observed* area of a star is $A = \pi R^2$ and the *observed* luminosity is $L\pi R^2$. Henceforth, we index quantities related to the primary star by 1 (e.g. primary mass is M_1) and 2 for the secondary star.

It is shown from Newton's laws that the orbits of an object in the gravitational field of another object is a conic section of eccentricity e . Here we are interested

in the case where $0 \leq e < 1$ that corresponds to closed orbits. We formulate two-body systems as systems where one body is fixed and the other is in orbital motion.¹

The position of the orbiting body is calculated by Kepler's equation as the distance r from the fixed companion star on the elliptical orbit [5],

$$r(t) = \frac{a(1 - e^2)}{1 + e \cos \theta(t)}, \quad (2)$$

where t is time and a is the semi-major axis of the ellipse calculated by Kepler's third law. Point Π in Fig. 1 is the periastron, the point where the distance between the orbiting and fixed body is minimal. Angle θ is the angle between the radius and the periastron. Knowledge of θ would allow us to determine the position of the orbiting body. Angle θ is indirectly inferred via an auxiliary circle centered at the center of the ellipse O and radius equal to semi-major axis. Point Q is the vertical projection of the orbiting body's position P to the auxiliary circle. Angle E is called the eccentric anomaly and is given by Kepler's equation [5]:

$$E(t) = e \sin E(t) + \frac{2\pi}{\rho}(t - \tau), \quad (3)$$

where τ is the instance of time that the body was at the periastron. Kepler's equation does not admit an analytical solution but can be approximated through the Newton-Raphson method. By geometrical arguments it is shown that the relation between the true and eccentric anomaly reads:

$$\tan \frac{\theta(t)}{2} = [(1 + e)/(1 - e)]^{\frac{1}{2}} \tan \left(\frac{E(t)}{2} \right) \quad (4)$$

By knowledge of θ we can determine the position of the second star on the orbit using (2) and (4). These positions correspond to the orbital plane and must be projected to the plane of the observer in the form of Cartesian coordinates [5]:

$$X(t) = r(t)(\cos(\Omega) \cos(\omega + \theta(t)) - \sin(\Omega) \sin(\omega + \theta(t)) \cos(i)), \quad (5)$$

$$Y(t) = r(t)(\cos(\Omega) \cos(\omega + \theta(t)) + \sin(\Omega) \sin(\omega + \theta(t)) \cos(i)), \quad (6)$$

$$Z(t) = r(t) \sin(\omega + \theta(t)) \sin(i), \quad (7)$$

which concludes the determination² of positions of the stars with respect to the observer.

An observer of the binary system receives a variable luminosity from the eclipsing binary system that plotted against time forms a light curve. This variability is due to the eclipses that occur when one body passes in front (in the

¹ It is shown in [6] that in the relative motion system, the eccentricity, period and semi-major of the moving body's orbit are equal to their counterparts in the two-body system, and only the masses transform.

² The angle Ω does influence the position of the orbiting body. However, it does not have an influence on the light curve and thus we treat it as a constant $\Omega = 0$.

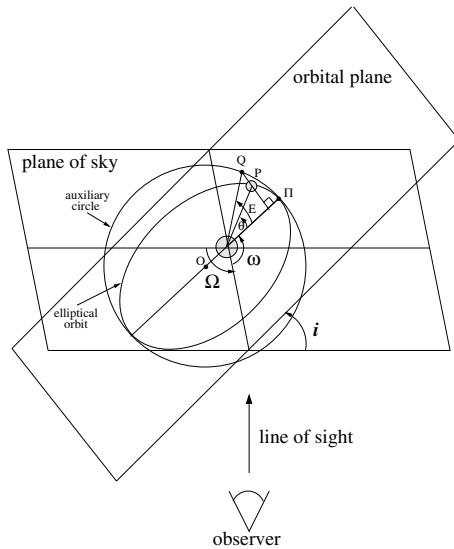


Fig. 1. Angles orientating the orbital plane with respect to the plane of sky, and angles associated with the orbits. Adapted from [5].

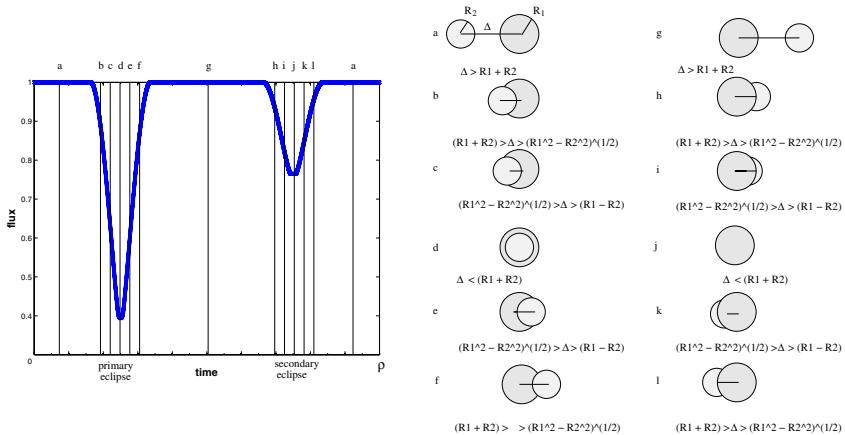


Fig. 2. Positions of stars (relative to observer's line of sight) and corresponding light curve phases

line of sight of the observer) of the other. This is illustrated in Fig. 2. When no eclipse occurs (positions *a*, *g*) the luminosity is equal to the sum of the luminosities radiated from the two bodies. The curved parts of the light curve occur due to partial occultations. Two eclipses take place at each period, one primary eclipse (position *d*), when the most luminous body of the pair is obscured the most, and a secondary eclipse (*j*), when the most luminous body obscures its companion the most.

Obscured parts of the disks of the stars can be calculated via geometrical arguments.³ The obscured area of each star at time t is denoted by $\Delta A_1(t)$ and $\Delta A_2(t)$. The luminosity $f_{\boldsymbol{\theta}}(t)$ received by the observer at time t depends on the luminosities L_i , areas A_i and obscured areas⁴ ΔA_i via

$$f_{\boldsymbol{\theta}}(t) = L_1(A_1 - \Delta A_1(t)) + L_2(A_2 - \Delta A_2(t)). \quad (8)$$

3 Noise Model for Light Curves

Based on the physical model a probabilistic generative noise model arises naturally. Observed light curves, denoted by \mathbf{O} , are noisy signals:

$$\mathbf{O}(t) = f_{\boldsymbol{\theta}}(t) + \epsilon(t), \quad (9)$$

where ϵ is i.i.d. Gaussian noise with variance σ^2 . Thus, we regard a light curve \mathbf{O} of period $\rho(\mathbf{O})$ sampled at times $t \in \mathcal{T} = \{t_1 = 0, t_2, \dots, t_T = \rho(\mathbf{O})\}$ as a realisation drawn from a multivariate spherical normal distribution. We denote the noise model associated with parameters $\boldsymbol{\theta}$ by $p(\mathbf{O}|f(\cdot; \boldsymbol{\theta}), \sigma^2)$ or simply by $p(\mathbf{O}|\boldsymbol{\theta})$.

4 Model for Topographic Organisation

The starting point of our model formulation is the form of a mixture model composed of C noise models as described in section 3.

$$p(\mathbf{O}|\boldsymbol{\Theta}) = \sum_{c=1}^C P(c) p(\mathbf{O}|\boldsymbol{\theta}_c), \quad (10)$$

where $P(c)$ are the mixing coefficients, $\boldsymbol{\Theta}$ encapsulates all parameter vectors $\{\boldsymbol{\theta}_c\}_{c=1:C}$ and $p(\mathbf{O}|\boldsymbol{\theta}_c)$ corresponds to the c -th model component with parameter vector $\boldsymbol{\theta}_c$. We simplify notation $p(\mathbf{O}|\boldsymbol{\theta}_c)$ to $p(\mathbf{O}|c)$. Assuming that dataset \mathcal{D} contains N independently generated fluxes $\mathbf{O}^{(n)}$, the posterior of the $\boldsymbol{\Theta}$ is expressed as:

$$p(\boldsymbol{\Theta}|\mathcal{D}) \propto p(\boldsymbol{\Theta}) \prod_{n=1}^N p(\mathbf{O}^{(n)}|\boldsymbol{\Theta}) = p(\boldsymbol{\Theta}) \prod_{n=1}^N \sum_{c=1}^C P(c) p(\mathbf{O}^{(n)}|c) \quad (11)$$

where the mixing coefficients can be ignored as $P(c) = \frac{1}{C}$.

Topographic organisation is introduced in the spirit of the GTM [1] by requiring that the component parameter vectors $\boldsymbol{\theta}_c$ correspond to a regular grid of points $\mathbf{x}_c, c = 1, \dots, C$, in the two dimensional latent space $\mathcal{V} = [-1, 1]^2$. A smooth nonlinear function Γ maps each point $\mathbf{x} \in \mathcal{V}$ to a point $\Gamma(\mathbf{x})$ that

³ See <http://www.physics.sfasu.edu/astro/ebstar/ebstar.html>. Last access on 12-0-07.

⁴ Recall that $i = 1$ and $i = 2$ index the primary and secondary stars, respectively.

addresses a model $p(\cdot|\mathbf{x})$. Points $\Gamma(\mathbf{x})$ are constrained on a two-dimensional manifold \mathcal{M} that is embedded in space \mathcal{H} , the space of parametrisations of our noise models. Since the neighbourhood of Γ -images of \mathbf{x} is preserved due to continuity of Γ , a topographic organisation emerges for the models $p(\cdot|\mathbf{x})$. Function Γ is realised as a RBF network [1]:

$$\Gamma(\mathbf{x}) = \mathbf{W}\phi(\mathbf{x}), \quad (12)$$

where matrix $\mathbf{W} \in R^{6 \times K}$ contains the free parameters of the model (6 is the number of parameters in $\{M_1, q, e, \iota, \omega, \rho\}$), and $\phi(\cdot) = (\phi_1(\cdot), \dots, \phi_K(\cdot))^T$, $\phi_k(\cdot) : R^2 \rightarrow R$ is an ordered set of K nonlinear smooth basis functions. However, this mapping may produce invalid parameter vectors, since the output of the RBF network is unbounded. We therefore redefine mapping Γ as:

$$\Gamma(\mathbf{x}) = \mathbf{A}g(\mathbf{W}\phi(\mathbf{x})) + \mathbf{v}, \quad (13)$$

where:

- g a vector-valued version of the sigmoid function that “squashes” each element in $[0, 1]$:

$$g(\mathbf{y}) = \left[\frac{1}{1 + \exp(-y_1)}, \frac{1}{1 + \exp(-y_2)}, \dots, \frac{1}{1 + \exp(-y_Y)} \right]^T, \quad (14)$$

- \mathbf{A} is a diagonal matrix that scales parameters to the appropriate range. \mathbf{A} has as diagonal elements the length of range $(\theta_i^{max} - \theta_i^{min})$ for each parameter, so that $A = diag((100 - 0.5), (1 - 0), (1 - 0), (2\pi - 0), (\frac{\pi}{2} - 0), (100 - 0.5))$.
- vector \mathbf{v} shifts the parameters to the appropriate interval. \mathbf{v} contains the minimum value θ_i^{min} for each parameter θ_i : $\mathbf{v} = [0.5, 0, 0, 0, 0, 0, 0.5]^T$.

The redefined mapping Γ now takes a point \mathbf{x} in space \mathcal{V} to a valid parameter vector $\Gamma(\mathbf{x})$ that addresses a noise model in \mathcal{M} . Thus, Θ has become a function of the weight matrix \mathbf{W} of the RBF network, $\Theta(\mathbf{W})$. Hence, the logarithm of the posterior from (11) now reads:

$$\log p(\Theta(\mathbf{W})|\mathcal{D}) \propto \log p(\Theta(\mathbf{W})) + \sum_{n=1}^N \log \sum_{c=1}^C p(\mathbf{O}^{(n)}|\mathbf{x}_c). \quad (15)$$

Figure 3 summarises the model formulation. Each point \mathbf{x} of the visualisation space \mathcal{V} is non-linearly and smoothly mapped via Γ to model parameters that identify the corresponding noise model $p(\cdot|\mathbf{x})$. These parameters are constrained on a two-dimensional manifold \mathcal{M} embedded in \mathcal{H} , the space of all possible parametrisations of our noise model. In the spirit of [1], the model can be used to visualise observed fluxes \mathbf{O} by calculating the posterior probability of each grid point $\mathbf{x}_c \in \mathcal{V}$, given \mathbf{O} :

$$p(\mathbf{x}_c|\mathbf{O}) = \frac{P(\mathbf{x}_c)p(\mathbf{O}|\mathbf{x}_c)}{p(\mathbf{O})} = \frac{P(\mathbf{x}_c)p(\mathbf{O}|\mathbf{x}_c)}{\sum_{c'=1}^C P(\mathbf{x}_{c'})p(\mathbf{O}|\mathbf{x}_{c'})} = \frac{p(\mathbf{O}|\mathbf{x}_c)}{\sum_{c'=1}^C p(\mathbf{O}|\mathbf{x}_{c'})}. \quad (16)$$

Each observed flux \mathbf{O} is then represented in the visualisation space \mathcal{V} by a point $proj(\mathbf{O}) \in \mathcal{V}$ given by the expectation of the posterior distribution over the grid points:

$$proj(\mathbf{O}) = \sum_{c=1}^C p(\mathbf{x}_c | \mathbf{O}) \mathbf{x}_c. \quad (17)$$

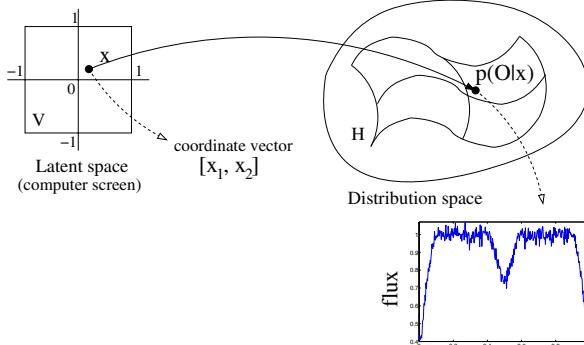


Fig. 3. Formulation of the topographic mapping model

We train our model in the MAP estimation framework with a physically motivated prior $p(\Theta)$ obtained from relevant literature [7,8,9,10]. To that purpose we employ the EM algorithm. Note that, due to the nature of the physical model formulation in sections 2 and 3, the M-step cannot be carried out analytically, nor can the derivatives of expected complete-data log-posterior with respect to the RBF network parameters \mathbf{W} be analytically obtained. However, the EM algorithm does not necessarily require that an optimum is achieved in the M-step; it is sufficient that the likelihood is merely improved [11]. For our purposes we resort to numerical optimisation by employing a (1+1) evolutionary strategy described in [12]. The fitness function for the evolutionary strategy is the expected complete-data log-posterior.

5 Experiments

5.1 Datasets

We performed experiments on two datasets. Dataset 1 is a synthetic dataset that consists of 200 light curves (fluxes). A common set of model parameters, $\{M_1 = 5, q = 0.8, e = 0.3, i = \frac{\pi}{2}\}$ was defined. However, two distinct values $\rho_1 = 2, \rho_2 = 5$ of period and $\omega_1 = 0, \omega_2 = \frac{5}{6}\pi$ of argument of periastron were used, to create 4 classes of light curves (50 in each class) by the combinations of these values, $\{\rho_1, \rho_2\} \times \{\omega_1, \omega_2\}$. The discerning characteristic of each class is the position of each secondary eclipse and the widths of the eclipses. Each

light curve was then generated from these four “prototypical” parameter settings corrupted by a Gaussian noise. Gaussian noise was also subsequently added to the generated light curves to simulate observational errors.

Dataset 2 consists of light curves from real observations obtained from two resources available⁵ on the WWW: the *Catalogue and Archive of Eclipsing Binaries* at <http://ebola.eastern.edu/> and the *All Sky Automated Survey*. Dataset 2 was preprocessed before training using local linear interpolations. Preprocessing is necessary as one needs to account for gaps in the monitoring process and for overlapping observations. Light curves must also be phase-shifted so that their first point is the primary eclipse and resampled to equal length as described in section 3. Finally, the light curves were resampled at $T = 100$ regular intervals which was judged an adequate sample rate.

5.2 Training

The lattice was a 10×10 regular grid (i.e. $C = 100$) and the RBF network consisted of $M = 17$ basis functions; 16 of them were Gaussian radial basis functions of variance $\sigma^2 = 1$ centred on a 4×4 regular grid in $\mathcal{V} = [0, 1]^2$, and one was a bias term. The variance of the observation noise in the local models $p(\mathbf{O}|\mathbf{x})$ was set to $\sigma^2 = 0.075$.

5.3 Results

Fig. 4 presents the topographic map constructed for the synthetic dataset. Each point stands for a light curve projected to latent visualisation space \mathcal{V} and is coloured according to class membership. The class memberships of synthetic fluxes were not used during the training process. Also, next to each cluster, a typical light curve has been plotted. The classes have been identified and organised appropriately, each occupying one of the four corners of the plot.

Fig. 5 presents the topographic map constructed for the dataset of real observed light curves. The red curves are the data projected against the underlying local noise models displayed in black. Several interesting observations can be made about the topographic formation of the light curves on the resulting map. In the lower right-hand corner binary systems of large periods are found. The median period of the systems in our sample is 2.7 days, and binaries like *V459 Cas*, with a period of 8.45 days lie in this corner. Systems with short period have the appearance of a wide V-shaped eclipse in the shape of their light curve, and inhabit the top and left edges of the map, e.g. *WY Hya* (Period: 0.7 days) and *RT And* (Period: 0.6 days). At the lower left of the map, we find systems with high eccentricity, e.g. *V1647 Sgr*. High eccentricity causes the light curve to appear assymetric, so that the period of the eclipse occurs further and further away from the center. On the other hand, very symmetric curves indicate orbits of low eccentricity (more circular) and low mass-ratio (stars of similar mass), and indeed we find systems like *DM Vir* ($e = 0.03$, mass ratio=1) and

⁵ Last accessed on the 12th September 2007.

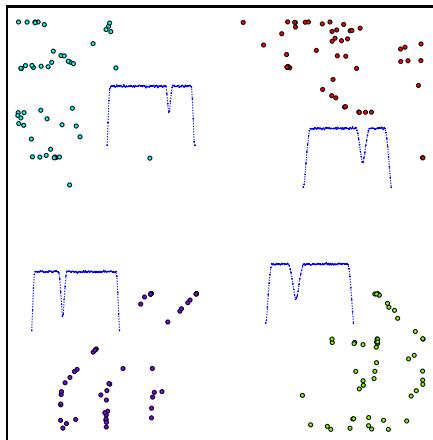


Fig. 4. Visualisation of synthetic dataset. A representative light curve is plotted next to each cluster.

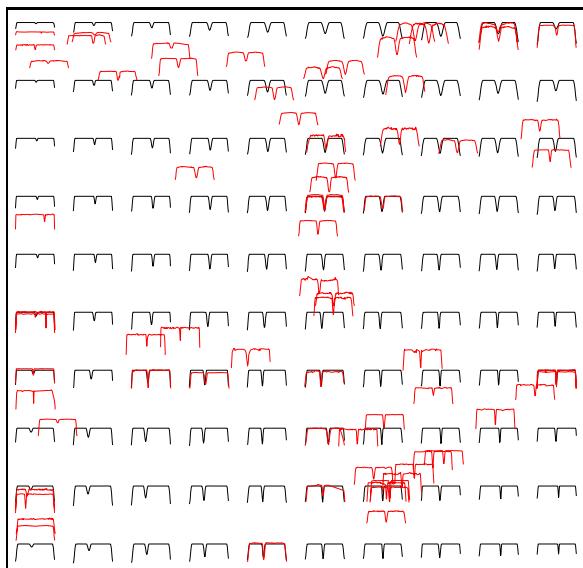


Fig. 5. Visualisation of dataset 2 of real data. Light curves in red are the projected real data and light curves in black are the light curves of the underlying local noise models.

CD Tau ($e = 0.0$, mass ratio=1.05) in the cluster in the lower-right hand corner of the map. Finally, low-inclination systems, occupy the top left-hand corner of the map, and these orbits will have very shallow eclipses as the companion star barely eclipses the primary star.

6 Conclusions

We have presented a model-based probabilistic approach for the visualisation of eclipsing binary systems. The model is formulated as a constrained-mixture of physically motivated noise models. As a consequence, a clear cost function naturally arises which drives the optimisation of the model. In our experiments we have demonstrated that the resulting maps can be interpreted in a transparent way by inspecting the underlying local noise models. Furthermore, modification and refinement of the local noise models is possible, to account for greater physical fidelity by incorporating physical aspects for non-spherical stars and even more sophisticated phenomena such as gravity darkening.

References

1. Bishop, C.M., Svensén, M., Williams, C.K.I.: GTM: The generative topographic mapping. *Neural Computation* 10(1), 215–234 (1998)
2. Kohonen, T.: The self-organizing map. *Proceedings of the IEEE* 78(9), 1464–1480 (1990)
3. Tiňo, P., Kaban, A., Sun, Y.: A generative probabilistic approach to visualizing sets of symbolic sequences. In: *KDD 2004: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–706. ACM Press, New York (2004)
4. Gianniotis, N., Tiňo, P.: Visualisation of tree-structured data through generative probabilistic modelling. In: Verleysen, M. (ed.) *European Symposium on Artificial Neural Networks, D-Facto*, pp. 97–102 (2007)
5. Hilditch, R.W.: *An introduction to close binary stars*. Cambridge University Press, Cambridge (2001)
6. Karttunen, H., Krger, P., Oja, H., Poutanen, M., Donner, K.J. (eds.): *Fundamental astronomy*. Springer, Heidelberg (1996)
7. Devor, J.: Solutions for 10,000 eclipsing binaries in the bulge fields of ogle ii using debil. *The Astrophysical Journal* 628(1), 411–425 (2005)
8. Halbwachs, J.L., Mayor, M., Udry, S., Arenou, F.: Multiplicity among solar-type stars. iii. statistical properties of the f7-k binaries with periods up to 10 years. *Astronomy and Astrophysics* 397, 159–175 (2003)
9. Miller, G.E., Scalo, J.M.: The initial mass function and stellar birthrate in the solar neighborhood. *Astrophysical Journal Supplement Series* 41, 513–547 (1979)
10. Paczyński, B., Szczygiel, D.M., Pilecki, B., Pojmański, G.: Eclipsing binaries in the All Sky Automated Survey catalogue. *Monthly Notices of the Royal Astronomical Society* 368, 1311–1318 (2006)
11. Ng, S., Krishnan, T., McLachlan, G.: The em algorithm. In: Gentle, J., Hardle, W., Mori, Y. (eds.) *Handbook of Computational Statistics*, vol. 1, pp. 137–168. Springer, Heidelberg (2004)
12. Rowe, J.E., Hidović, D.: An evolution strategy using a continuous version of the gray-code neighbourhood distribution. In: Deb, K., et al. (eds.) *GECCO 2004, Part I. LNCS*, vol. 3102, pp. 725–736. Springer, Heidelberg (2004)

Generalized Self-Organizing Mixture Autoregressive Model for Modeling Financial Time Series

Hujun Yin¹ and He Ni²

¹ School of Electrical & Electronic Engineering, University of Manchester,
Manchester, UK

² School of Finance, Zhejiang Gongshang University, HangZhou, China
h.yin@manchester.ac.uk, nihe@mail.zjgsu.edu.cn

Abstract. The mixture autoregressive (MAR) model regards a time series as a mixture of linear regressive processes. A self-organizing algorithm has been used together with the LMS algorithm for learning the parameters of the MAR model. The self-organizing map has been used to simplify the mixture as a winner-takes-all selection of local models, combined with an autocorrelation coefficient based measure as the similarity measure for identifying correct local models and has been shown previously being able to uncover underlying autoregressive processes from a mixture. In this paper the self-organizing network is further generalized so that it fully considers the mixing mechanism and individual model variances in modeling and prediction of time series. Experiments on both benchmark time series and several financial time series are presented. The results demonstrate the superiority of the proposed method over other time-series modeling techniques on a range of performance measures including mean-square-error, prediction rate and accumulated profits.

Keywords: Time series, mixture autoregressive model, self-organizing map.

1 Introduction

Financial analysis relies greatly on time series models. Linear regression and autoregressive models are among the widely used methods in time series modeling. Most existing linear models developed in statistics and signal processing assume that the time series being dealt with is stationary and uni-modal [3, 8]. Such conditions are not often met in practical applications. For instance, autoregressive (AR), moving average (MA) and autoregressive moving average (ARMA) models assume a structured linear relationship between the current value of the time series and its previous values and the error terms. The relationship is described by a set of constant coefficients. This assumption is a pitfall of regressive models when the time series is non-stationary. Developing methods for modeling non-stationary, nonlinear and multimodal time series has become an active

area of research in statistics, signal processing, control systems and neural networks. The autoregressive integrated moving average (ARIMA) [3], a more generalized ARMA model, can better deal with slow changing non-stationary time series by modeling the difference of the consecutive time series values instead of the value itself. The generalized autoregressive conditional heteroscedastic (GARCH) model [2] describes the variance of current residual as a linear function of the variances of previous residuals, in addition to autoregressive model of the time series. It has been widely applied to modeling financial time series, which exhibit different volatilities from time to time. The recent mixture autoregressive (MAR) model [17] represents another valid approach. It considers the process as a mixture of regressive models and is a generalized Gaussian mixture transition distribution. The MAR model can handle non-stationary cycles and conditional heteroscedasticity and is often solved using the expectation-maximization (EM) method.

Many adaptive neural network models have been adopted or used to extend linear regressive models, such as multilayer perceptron (MLP), radial basis function (RBF) networks, support vector machines (SVM) and recurrent networks (see [9]). For instance, RBFs have been used to construct the NARMAX (non-linear autoregressive moving average with exogenous inputs) model in systems identification [5]. The self-organizing map (SOM) has also been applied to model time series in a variety of ways. In an indirect way, the SOM can be used to divide or cluster a time series into segments. A number of regressive models can be created by locally fitting to the corresponding parts of the time series. For instance, Dablemont et al. [6] applied SOM-based local models with RBF networks as regressors to predict the returns of stocks. Cao [4] proposed a SVM expert system, which is also based on SOM-clustered local models, to predict the future movement of a time series. However, these models are two-stage models. Both clustering and local modeling may not be jointly optimized.

There are two early approaches to analyzing temporal or sequence signals with the SOM. One is to train a SOM on static states (i.e. time series values), and then temporal patterns or sequences of states can be identified by marking sequential locations of the states on the trained map. Such approaches can be used to monitor dynamic processes or trajectories of a temporal process such as an industrial plant, e.g. [1]. Another approach, which is often found in the literature, is to group consecutive time points into segments (using a sliding window). Then these segments are used as the input vectors to train the SOM. We term this method as vector SOM (VSOM). Although consecutive time points can be grouped into vectors to form temporal context by means of a sliding window, the vectors formed may not capture temporal relationship of the time series. Several variants have since been proposed to extend SOM's ability for temporal modeling such as, the recurrent SOM (RSOM) [10] and the recursive SOM (RecSOM) [16] or merge SOM (MSOM) [15]. These SOM variants integrate the information of a sequence via recursive operations. As they differ in the notion of context, their efficiency in terms of representing temporal context are different. Neural gas [12] is another popular variant of SOM. Instead of having

a fixed network topology throughout, neural gas can dynamically deploy its resources to suit varying topology of the data, and has been applied to many tasks including vector quantization and temporal modeling [12].

Earlier, Lampinen and Oja proposed a self-organizing map of spatial and temporal AR models (SOAR) [11], where each unit represents an AR model with its reference vector as the model parameters. The experiments have shown that the model can learn to distinguish textures in images. The method in fact is a multiple AR model with the component models forming a spatial topology. However, the model has difficulties to converge to the underlying regressive processes due to the simple error-based similarity measure. We have extended the SOAR model to a mixture regressive model, termed self-organizing mixture autoregressive (SOMAR) model [13], with a different partition mechanism and similarity measure to reflect the characteristics of homogeneous time series. Both the mixture and local models are jointly trained, and thus it offers better modeling performance. Good results have been reported [13, 14]. In this paper the SOMAR model is further generalized to a full mixture model, with parameters learnt by a self-organizing algorithm. Section 2 briefly describes existing regressive time series models. Section 3 presents SOM-based autoregressive models and the proposed generalized SOMAR (GSOMAR) model, followed by experimental results on both benchmark data and real-world financial data and comparisons with existing methods in Section 4. Finally, conclusions are given in Section 5.

2 Regressive Time Series Models

The widely used time series models include AR, MA and ARMA and nonlinear or non-stationary ARIMA, GARCH and MAR models. The MAR is the theoretical basis of the proposed model. The solutions of the AR, MA, ARMA, GARCH and ARIMA models can be found in many time series and signal processing books.

2.1 Autoregressive Models: AR, ARMA, GARCH and ARIMA

Linear regressive models have been the primary tool in modeling time series. An autoregressive model of order p , denoted as AR(p), can be described as,

$$x_t = c + \sum_{i=1}^p \phi_i x_{t-i} + \varepsilon_t = c + \Phi^T \mathbf{x}_{t-1}^{(p)} + \varepsilon_t \quad (1)$$

where $\Phi = [\phi_1, \dots, \phi_p]^T$ are the model parameters, $\mathbf{x}_{t-1}^{(p)} = [x_{t-1}, \dots, x_{t-p}]^T$ is the concatenated input vector, c is a constant and ε is white noise with zero mean and variance σ^2 .

An ARMA model with p -order AR terms and q -order MA terms is called ARMA(p, q) model and can be written as,

$$x_t = c + \sum_{i=1}^p \phi_i x_{t-i} + \sum_{i=0}^q \mu_i \varepsilon_{t-1-i} \quad (2)$$

where $\{\mu_0, \dots, \mu_q\}$ are the parameters of the moving average. The error terms are assumed to be independent identically-distributed (i.i.d.) random variables sampled from a normal distribution with zero mean and constant variance σ^2 , which is often associated with the volatility of the time series. When this condition does not hold as it is often the case in financial data, the GARCH model provides a generalized alternative, in which the variance of the error terms is modeled by another autoregressive model.

A standard GARCH(θ, ρ) model is characterized by Eq. (1) and the following variance model,

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{\theta} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=0}^{\rho} \beta_i \sigma_{t-i}^2 \quad (3)$$

where ε_t is the error term with the assumption $\varepsilon_t = \sigma_t v_t$ and v_t is i.i.d. with zero mean and unit variance. $\{\alpha\}$ and $\{\beta\}$ are the model parameters of the variance.

ARIMA model uses lags or differencing of the time series in the ARMA model. ARIMA(p, d, q) model is characterized by the following equation,

$$(1 - \sum_{i=1}^p \phi_i L^i)(1 - L)^d x_t = (1 + \sum_{i=1}^q \mu_i L^i) \varepsilon_t \quad (4)$$

where L is lag operator, i.e. $Lx_t = x_{t-1}$ and p, d and q are the orders of the autoregressive, integrated, and moving average parts of the model respectively. Note that ARMA(p, q), i.e. Eq.(1), can be expressed as $(1 - \sum_{i=1}^p \phi_i L^i)x_t = (1 + \sum_{i=1}^q \mu_i L^i)\varepsilon_t$.

As can be seen, the ARIMA model operates on the difference of the lagged time series. Such simple transformation can be effective in dealing with slow (seasonal) changes in non-stationary series. That is, the difference operator transforms a slow drift non-stationary process into a stationary process.

2.2 Mixture Autoregressive (MAR) Model

A nonlinear or non-stationary time series can be regarded as a mixture of stationary processes characterized by the standard autoregressive models. The K-component MAR model is defined by [17],

$$F(x_t | \Gamma_{t-1}) = \sum_{k=1}^K \pi_k \varphi\left(\frac{x_t - \phi_{k0} - \phi_{k1}x_{t-1} - \dots - \phi_{kp_k}x_{t-p_k}}{\sigma_k}\right) \quad (5)$$

where $F(x_t | \Gamma_{t-1})$ is the conditional distribution of x_t given the past information up to $t-1$, Γ_{t-1} ; $\varphi(\cdot)$ is the standard normal distribution; $\{\pi_1, \dots, \pi_K\}$ are the mixing parameters and $\pi_1 + \dots + \pi_K = 1$, $\pi_k > 0$, $k = 1, \dots, K$; p_k is the order of the k -th AR model; and σ_k^2 is the variance of the k -th distribution. This model is denoted as MAR($K; p_1, \dots, p_K$) model. It has been reported to have the ability to handle cycles and conditional heteroscedasticity in time series. Its parameters are estimated via the EM algorithm and model selection and validity can be done by a Bayesian information criterion (BIC) [17].

3 Self-Organizing Mixture Autoregressive Models

The proposed model is an extension of an earlier model by Lampinen and Oja and further enhanced in the framework of the MAR model.

3.1 Self-Organizing AR (SOAR) Model

In 1989, Lampinen and Oja proposed a self-organizing AR (SOAR) network [11]. It is a map of “neural” units where each unit i represents an AR model with its parameters as the reference vector w_i . The experiment shows that the SOAR model can learn to distinguish texture images by unsupervised learning [11]. The method in fact is a multiple AR model. However, the model has difficulties in converging to the correct AR models. The procedure of the SOAR model is:

1). At each time step t , find the best matching unit by measuring the estimation error of each node, $e_{i,t} = x_t - \mathbf{w}_i^T \mathbf{x}_{t-1}^{(p)}$. In order to reduce the effect of the fluctuation or noise in the errors, an exponential average over recent estimation errors is used,

$$u_{i,t} = \lambda e_{i,t} + (1 - \lambda) u_{i,t-1} \quad (6)$$

where λ is a smoothing factor, $e_i(t)$ is the current error of node i and $u_i(t-1)$ is the past averaged error.

2). Update the best matching unit as well as the units in its map neighborhood, by the recursive LMS or Widrow-Hoff rule,

$$\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1} + \eta h(v, i) e_{i,t} \mathbf{x}_t^{(p)} \quad (7)$$

where η is learning rate and $h(i, v)$ is the neighborhood function of indexes of node i and winner v .

However, the performance of the SOAR model in finding the underlying AR processes in the mixture is poor [13, 14]. Due to the stochastic nature of AR processes, although the overall MSE decreases, at each input one can always expect large oscillation even when the true model parameters are used or further manipulations, e.g. exponential smoothing, are applied. In other words, this method has difficulties to converge to the true model parameters of the underlying AR processes. Nevertheless, the SOAR model localizes the time series by local models.

3.2 Self-Organizing MAR (SOMAR) Model

Based on the similar self-organizing principle and the MAR model, the self-organizing MAR (SOMAR) was proposed by using the SOM and LMS algorithm to learn a simplified MAR model [13]. The SOMAR constitutes a simplified MAR model with the winner-takes-all principle for local AR models. To ensure a robust learning, a new winner selection or similarity measure was proposed. A stochastic process is characterized by white noise residuals. As a sufficient condition, the

modeling errors or residuals should be or close to white noise if the modeling is following the correct path. Therefore, in the SOMAR model, the autocorrelation of the error instead of the error itself is used to evaluate the similarity between the input vector and the neurons' weights representing the model parameters. In order to have a reliable estimate of autocorrelation, a small batch of the errors is used,

$$R_{i,t}(k) = \frac{1}{m\sigma_e^2} \sum_{l=0}^{m-p-1} (e_{i,t-l} - \mu_e)(e_{i,t+k-l} - \mu_e) \quad (8)$$

where m is the length of the batch, μ_e and σ_e^2 are the mean and the variance of the errors in the batch respectively.

The winner is selected according to the sum of (absolute value of) autocorrelation coefficients (SAC),

$$v = \arg \min_i \left(\sum_{k=-m}^m |R_{i,t}(k)| \right) \quad (9)$$

The use of correlation measure for identifying local models is justified by the fact that a correct model produces white noise residuals. That is, if the model is correct or adequate, the residual is unpredictable or structure-less. Such effective correlation-based tests are often used in statistics and neural networks for checking the fitness of a model, e.g. [5], although there are other whiteness tests in the literature.

3.3 Generalized SOMAR Model

Both SOMAR and SOAR models represent a simplified, homoscedastic and winner-takes-all version of the MAR model. At any particular time, usually only one local AR model (the winner) is selected to represent the time series, all AR models are assumed of equal variance and the mixing factors are either unit for the winner or zero otherwise (winner-takes-all). Although some empirical use of local neighboring nodes have been proposed for forecasting [13, 14], these models are not a full mixture model. To fully employ the mixture model, all components will be required to contribute to the mixture coherently, both in training and testing. The mixing factors and model variances have to be learnt as well. The SOM has been extended before to represent a mixture model. The self-organizing mixture network (SOMN) [18] is such an example, in which each node represents a conditional distribution. The SOMN has been shown to converge faster and be more robust than the EM algorithm for heteroscedastic mixture distributions. Therefore, to make the SOMAR a fully mixture of AR models, the learning algorithm of the SOMN can be used for learning the mixing factors and model variances. We term this enhanced SOMAR a generalized self-organizing mixture autoregressive (GSOMAR) model. In addition to the weights (or parameters of component AR models), the mixing factors and model variances are updated. Further assume that the component models are uncorrelated, so that their

covariances are zeros. The variances of local models are scalar. Then based on the SOMN algorithm [18], the updating rules for the mixing weights and variances have the following simple forms,

$$\pi_{i,t} = \pi_{i,t-1} + \eta(\hat{P}_i - \pi_{i,t-1}) \quad (10)$$

$$\sigma_{i,t}^2 = \sigma_{i,t-1}^2 + \eta(\sigma_{i,e}^2 - \sigma_{i,t-1}^2) \quad (11)$$

where \hat{P}_i is the winning frequency and $\sigma_{i,e}^2$ is the error variance of node i .

The trained mixture model, representing the MAR model, Eq. (5), can be fully used for forecasting the time series as well as model's volatility. In forecasting, the learnt mixing factors are further weighted by the neighborhood function of the SOM acting as the posterior probability of a component class given an input sample [18].

4 Experimental Results and Comparisons

Experiments on bench mark Mackey-Glass time series and several financial data sets were conducted, together with comparisons with related methods and models.

4.1 Mackey-Glass Data

The Mackey-Glass series has been widely used as a benchmark data for testing nonlinear models. The data set was generated by a dynamic system defined by the following differential equation,

$$\frac{dx}{dt} = \beta x_t + \frac{\alpha x_{t-\delta}}{1 + x_{t-\delta}^{10}} \quad (12)$$

with the parameter values set as $\delta = 17$, $\alpha = 0.2$, and $\beta = -0.1$. In total 2000 points were generated. The Mackey-Glass data is regarded as consisting of a number of unknown AR processes. The order of the AR processes was chosen by the BIC. In the experiment, the series was grouped into 12 consecutive values as the input vectors. The prediction result is shown in Table 1. Both the SOMAR and GSOMAR models have outperformed other models; while GSOMAR also improves on the SOMAR markedly.

Table 1. Forecasting performance on Mackey-Glass data by various adaptive models

	VSOM	RSOM	RecSOM	MSOM	NeuralGas	SOM +SVM	SOAR	SOMAR	GSOM- AR
MSE($^{-2}$)	4.52	4.34	4.19	4.31	4.39	4.52	4.30	3.62	3.25

4.2 Foreign Exchange Rates

The data set was obtained from the PACIFIC Exchange Rate Service provided by W. Antwiler at UBCs Sauder School of Business. It consists of 15 years' daily exchange rates (of three pairs: British pound vs. US dollar, Euro and HK dollar, respectively) excluding weekends and bank holidays when currency markets were closed. In total 3200 consecutive points were used, in which the first 3000 points were used as the training set, the next 100 points as the validation set, and the remaining 100 points as the test set. The training, validation and testing sets were windowed with the length of 15 points to form input vectors. The split of the data set was chosen in favor of short range forecasting. Other splits have also been tested and results vary slightly. To compare with other regressive models in financial data modeling and forecasting, the following commonly used performance measures have been evaluated:

Predicted return(%): The correct prediction percentage of the return ($\ln \frac{x_{t+1}}{x_t}$), which is also used as a criterion to check whether the prediction is made in the right direction. In other words, it shows how many percentages of the predicted returns have the same signs as their corresponding actual returns.

MSE of predicted rate ($^{-2}$): The MSE between the actual exchange rates and the predicted ones in the test set.

Accumulated profit(P%): The accumulated profit is the percentage gain of the accumulated profits over the testing period, say 100 trading days.

Table 2. Performance on FX rate prediction by various regressive models. The best performances are marked in bold.

GBP	VSOM	RSOM	RecSOM	MSOM	Neural	SOM+	SOAR	SOMAR	GSOMAR
vs					Gas	SVM			
USD %	52.63	52.26		52.58	52.54	54.09	53.43	52.84	59.72
USD $^{-2}$	4.20	4.24		4.70	4.52	4.24	4.12	4.28	3.80
USD P%	4.80	4.98		5.12	4.92	5.32	4.82	4.78	5.16
EU %	52.12	53.05		53.17	53.21	54.24	54.09	52.62	56.42
EU $^{-2}$	4.32	4.64		4.95	4.52	4.50	4.62	4.73	4.10
EU P%	4.73	4.63		4.60	4.72	4.74	4.70	4.62	5.13
JPY %	54.29	52.48		52.33	52.40	53.45	52.10	53.22	57.31
JPY $^{-2}$	5.00	4.89		5.08	5.00	4.75	5.18	5.24	4.31
JPY P%	4.89	4.91		4.87	4.87	4.72	4.65	4.69	5.01

As reported before [13,14], the SOMAR model generally outperforms other methods as also shown in Table 2. The GSOMAR further improves on the SOMAR model in all these performance measures. As can be seen, both GSOMAR and SOMAR consistently outperform other methods by clear margins in the correct prediction percentages and modeling errors. The benefit of using the fuller

Table 3. Performance on stock index modeling by various regressive models. The best performances are marked in bold.

	VSOM	RSOM	RecSOM	MSOM	Neural Gas	SOM+	SOAR	SOMAR	GSOMAR
					Gas		SVM		
DJ %	53.08	53.36	53.24	54.21	54.67	53.67	52.64	55.99	56.69
DJ $^{-2}$	8.02	7.89	7.86	7.79	7.72	8.10	8.23	7.52	7.23
FT %	53.20	53.87	53.79	54.01	54.29	53.76	53.38	56.10	56.81
FT $^{-2}$	7.90	7.77	7.78	7.65	7.54	7.80	7.89	7.24	6.98
SP %	54.71	55.24	55.34	55.87	56.03	54.90	54.78	56.77	57.12
SP $^{-2}$	6.34	6.37	6.32	6.38	6.21	6.40	6.31	5.98	5.56
HS %	53.98	53.78	54.10	54.21	54.57	53.29	54.28	56.01	56.35
HS $^{-2}$	7.01	6.82	6.79	6.83	6.61	7.35	6.89	6.39	5.87

GSOMAR model is that model variance parameters are readily available to indicate the volatility of the component regressive models and the mixture. The length of the input or local model order was validated as 15 and patch size was chosen as 8 in this experiment. Further tests on a range of stock indices such as Dow-Jones (DJ), FTSE(FT), S&P (SP) and Hang Seng (HS) are also shown in Table 3. The data (daily closing values during 1986-2009) was obtained from Yahoo Finance, and was adjusted by including the effects of dividends and splits. The most recent 3200 data points were used in the experiments. These results further demonstrate the improved performance by the GSOMAR model and the benefit of generalizing to a fuller mixture autoregressive model in financial time series modeling. Both SOMAR and GSOMAR models have produced markedly better performance in terms of prediction percentages and mean-square errors. Note that no accumulated profit is available in this case as the stock indices are not directly tradable.

5 Conclusions

A mixture model approach to tackling nonlinear and non-stationary time series has been proposed by using a generalized self-organizing mixture autoregressive (GSOMAR) model. It consists of a number of autoregressive models that are organized and learned in a self-organized manner by the adaptive LMS algorithm. A correlation-based similarity measure is used for identifying the correct AR models, making the model more effective and robust compared to the error-based measures. The GSOMAR further generalizes the winner-takes-all SOMAR model by learning the mixing weights as well as the model variances. These parameters enhance the simplified SOMAR model and improve the modeling accuracy. The experiments have been conducted on various nonlinear, non-stationary processes including financial time series. The results show that the proposed method outperforms other SOM-based methods in terms of modeling errors and prediction performances on several modeling and forecasting performance measures.

References

1. Allinson, N.M., Yin, H.: Interactive and semantic data visualization using self-organizing maps. In: Proc of IEE Colloquium on Neural Networks in Interactive Multimedia Systems (1998)
2. Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31, 307–327 (1986)
3. Box, G., Jenkins, G.: *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco (1970)
4. Cao, L.J.: Support vector machines experts for time series forecasting. *Neurocomputing* 51, 321–339 (2002)
5. Chen, S., Billings, S.A., Cowen, C.F.N., Grant, P.M.: Practical identification of NARMAX models using radial basis functions. *Int. Journal of Control* 52, 1327–1350 (1990)
6. Dablemont, S., Simon, G., Lendasse, A., Ruttens, A., Blayo, F., Verleysen, M.: Time series forecasting with SOM and local non-linear models - Application to the DAX30 index prediction. In: Proc. of WSOM 2003, pp. 340–345 (2003)
7. Kohonen, T.: *Self-Organizing Maps*. Springer, Heidelberg (1997)
8. Enders, W.: *Applied Econometric Time Series*, 2nd edn. John Wiley & Sons, Chichester (2004)
9. Haykin, S.: *Neural Networks - A Comprehensive Foundation*, 2nd edn. Prentice-Hall, Englewood Cliffs (1998)
10. Koskela, T.: Time Series Prediction Using Recurrent SOM with Local Linear Models, Helsinki University of Technology (2001)
11. Lampinen, J., Oja, E.: Self-organizing maps for spatial and temporal AR models. In: Proc. of 6th SCIA Scandinavian Conference on Image Analysis, Helsinki, Finland, pp. 120–127 (1989)
12. Martinet, T., Berkovich, S., Schulten, K.: Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Trans. Neural Networks* 4, 558–569 (1993)
13. Ni, H., Yin, H.: Time-series prediction using self-organizing mixture autoregressive network. In: Yin, H., Tino, P., Corchado, E., Byrne, W., Yao, X. (eds.) IDEAL 2007. LNCS, vol. 4881, pp. 1000–1009. Springer, Heidelberg (2007)
14. Ni, H., Yin, H.: Self-organizing mixture autoregressive model for non-stationary time series modeling. *Int. Journal of Neural Networks* 18, 469–480 (2008)
15. Strickert, M., Hammer, B.: Merge SOM for temporal data. *Neurocomputing* 64, 39–72 (2005)
16. Voegtlin, T., Dominey, P.F.: Recursive self-organizing maps. *Neural Networks* 15, 979–991 (2002)
17. Wong, C.S., Li, W.K.: On a mixture autoregressive model. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* 62 (part1), 95–115 (2000)
18. Yin, H., Allinson, N.M.: Self-organizing mixture networks for probability density estimation. *IEEE Trans. on Neural Networks* 12, 405–411 (2001)

Self-Organizing Map Simulations Confirm Similarity of Spatial Correlation Structure in Natural Images and Cortical Representations

A. Ravishankar Rao and Guillermo Cecchi

IBM T.J. Watson Research Center,

Yorktown Heights, NY 10598, USA

ravirao@us.ibm.com, gcecchi@us.ibm.com

Abstract. The goal of this paper is to extend the ecological theory of perception to the larger scale spatial organization of cortical maps. This leads to the hypothesis that cortical organization of responses to visual features reflects the environmental organization of these same features. In our previous work we have shown that the spatial statistics of natural images can be characterized by a slowly decaying, or low frequency correlational structure for color, and a rapidly decaying, or high-frequency structure for orientation features. A similar contrasting behavior of spatial statistics for color and orientation was measured in parallel in the cortical response of macaque visual cortex.

In order to explore whether this parallel is meaningful, we performed a cortical simulation using an adaptation of Kohonen’s self-organizing map algorithm. The simulated cortex responds to both low-frequency and high-frequency input visual features, and learns to represent these features through weight modification. We demonstrate that the learnt cortical weights show the same spatial correlation structure that is observed both in natural image statistics and the measured cortical responses.

1 Introduction

According to evolutionary theory, all organisms adapt to their environment, reaching a state where an organism functions optimally in its environment. This theory can also be applied to organs, where their form and function can be considered to optimize the survival of the organism in its environment. There is evidence that the brain has evolved similarly, such that “the better the ‘matching’ between brain, body and environment, the better the adaptation of the organism to its respective environment.” [16][p. 344].

This motivates an ecological approach to perception, which explores environmental influences on the structure of the sensory apparatus. This approach was used to derive retinal ganglion spectral responses based on spectral properties of natural scenes [1]. However, the cortical representations of natural images and their features such as color, orientation and spatial frequency have proved harder to measure and characterize. Recent evidence shows that horizontal and vertical

orientations are more likely to occur in natural scenes, and that there are more cortical neurons that are tuned for these orientations than others [13]. Hence, an open question is: what is the relation between statistics of natural image features, and the statistics of their cortical representations? Our interpretation of the ecological theory is that these statistics should be similar. Indeed, early research in this direction suggests that this is the case [7].

In order to explore this hypothesis further, we can also apply modeling techniques to determine how different statistical distributions of input features are manifested in the cortical representations of those features. Are they comparable, or dissimilar? The approach taken in this paper is illustrated in Figure 1. The observations drawn from such a model could independently verify biological observations, and serve to explain cortical functioning and processing. This is especially valuable when obtaining the necessary biological observations is difficult, say due to obtaining measurements from multiple subjects due to cost, or due to the complexity of the experimental protocol.

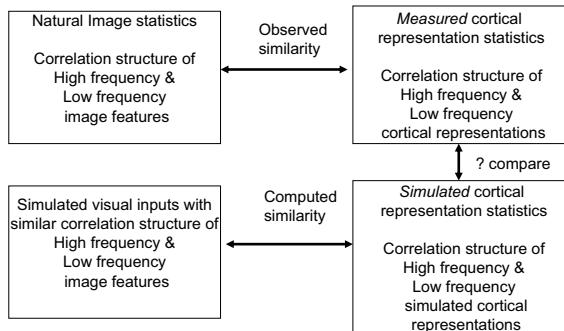


Fig. 1. The modeling approach used in this paper. We wish to compare the cortical representation statistics in measured samples with simulated samples.

We use an adapted Kohonen self-organizing map based on our previous work [18] to model cortical functioning. The Kohonen algorithm [12] uses a local neighborhood to smooth weights, resulting in similarity in cortical responses over local neighborhoods in general. However, an open question is whether it preserves higher order structures such as longer range correlations that may be present in the inputs. We argue that preservation of longer range correlations in the cortical representation is essential so that higher cortical levels (say V2 and higher) can efficiently read out information from lower level cortical maps. For instance, contour completion tasks [14] can be carried out more easily if longer range input correlations are preserved at the level of cortical area V1. Thus is it advantageous for the spatial structure of the cortex to capture both local correlations and higher order correlations of inputs.

Our simulation results indicate that the spatial correlation structure of cortical responses to high and low frequency input features mirrors the correlation

structure of these features in the input space. In other words, there is a correspondence between the input space and the cortical representation.

The implications of our result are that the hypothesis of ecological perception is valid, and can be extended to sensory areas of the brain. Furthermore, our findings can inspire neuroscientists to make measurements that would provide additional evidence. Since the number of possible neuroscience experiments is vast, our research suggests specific measurements that need to be made to advance the ecological theory of perception. This could prove to be a useful instance of computer modeling driving neuroscientific observation. Mathematical and computational models can play a vital role here, by providing testable predictions that can guide effective experimentation.

2 Background

Atick and Redlich [1] derived the spatial receptive field properties of retinal ganglion cells as an optimal filter for the two-point correlations of contrast in natural images. Since nearby pixels are highly correlated, this implies that nearby retinal receptors are highly redundant [21,20], and need not transmit their information in full. This is a demonstration of the application of the ecological theory of perception to the functioning of the sensory apparatus.

However, the interpretation of cortical functioning and organization according to this theory has proved more difficult. A striking property of visual cortical areas is the topographic organization of features such as orientation, color, and spatial frequency, whereby nearby units tend to display similar feature preferences [10]. Sejnowski and Bell show that oriented edge detectors are the maximal information filters for natural images [3], but these filters cannot be spatially organized on the basis of the similarity of their responses. Indeed, cortical map clustering seems to preserve redundancies rather than reduce them, at least when the average firing rate is considered as the coding response.

Self-organized feature maps, on the other hand, have proved to be a more amenable model of the visual cortex. Kohonen's self-organized algorithm has been used to model map formation, under the principle that the topology of the input space should be preserved in the topology of neural space [17]. Maps trained with oriented stimuli recreate to a large extent the spatial organization of primary visual cortex [17], including selectivity and linear zones punctuated by point and line discontinuities in the case of orientation, and a biologically consistent spatial layout of color selectivity [19]. Self-organized maps also have the advantage of a relatively simple physiological interpretation [15]. Moreover, in a seemingly significant departure from the ecological theory, these maps allocate more neural space to more frequent inputs [17], as opposed to equalizing them with the more infrequent ones.

Therefore the following questions are pertinent: what is the relationship between the *statistics of natural images* and the *spatial structure of cortical maps*? Is it possible to identify a functional dependence of cortical response maps of orientation and color with the ensemble properties of the corresponding inputs?

2.1 Statistics of Natural Images

Scaling measurements involve studying how the probability of finding a cooccurring pair changes as a function of the relative distance. A classic result in the analysis of natural scenes is that the luminance of pairs of pixels is correlated and that this correlation is scale-invariant [9,21,1]. This indicates that statistical dependencies between pairs of pixels do not depend on whether the observer zooms in on a small window or zooms out to a broad vista.

Few studies have focused on the structure of long-range correlations of other visual attributes. A report by Sigman *et al.* [22], analyzed the spatial distribution of orientation information in natural scenes. They showed that information about the presence of iso-oriented lines in natural images is correlated over relatively short distances, following specific power-law statistics for co-linearity; other pairwise arrangements display shorter correlations. Cecchi *et al.* [2] presented a preliminary study showing that the correlation structure in natural images shows a slowly varying, low frequency characteristic for color, and a faster varying, high frequency characteristic for orientation. This relationship was preserved in the correlation structure of cortical responses to color and orientation features in the input space.

Betsch *et al.* [5] showed that the environment is dominated by the cardinal horizontal and vertically oriented segments. Li *et al.* [13] measured the orientation tuning characteristics of a population of several thousand cat striate cortical neurons and found that more neurons prefer the cardinal rather than oblique angles. Betsch *et al.* [5] suggest that the increased cortical representation of cardinal orientations may be a natural consequence of image statistics. This shows that the cortical representation of visual attributes is strongly related to natural image statistics and is a research direction worth exploring.

2.2 Self-Organizing Maps

Kohonen's self-organizing map (SOM) has been widely used in a number of domains [12]. It has had considerable impact in computational neuroscience through the modeling of cortical map formation [8,2]. In our earlier work [18] modified Kohonen's self-organizing map (SOM) algorithm to employ only local competition, and developed a technique to eliminate the traditional parameterizations required [4].

The role of inputs is critical in the process of self-organization. Hubel *et al.* [11] showed that rather than being genetically predetermined, the structure of cortical visual area V1 undergoes changes depending on the animal's visual experience, especially during the critical period of development.

Certain classes of inputs are sufficient to model V1. For instance, Bednar [2] used input stimuli consisting of elongated Gaussian blobs. In this paper, we use sine-wave gratings of random orientation for the sake of simplicity, and to demonstrate the essential characteristics of our solution.

3 Experimental Methods

We model the visual pathway from the retina to the cortex as shown in Figure 2. The retina projects to the lateral geniculate nucleus (LGN), which in turn projects to the cortex. There are two channels in the LGN, which perform on-center and off-center processing of the visual input. The cortical units are interconnected through a lateral network which is responsible for spreading the weights of the winner.

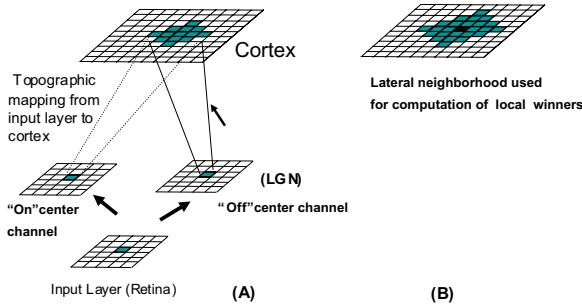


Fig. 2. Illustrating the network connectivity. (A) The input units are arranged in a two-dimensional grid, and can be thought of as image intensity values. The cortical units also form a 2D grid. Each input unit projects via the LGN in a feedforward topographic manner to the cortical grid. (B) shows the lateral connectivity in the cortex.

3.1 Algorithm for Weight Updates

We have extended Kohonen's algorithm to allow local competition in a larger cortex, such that multiple winners are possible [18]. Learning is driven by winners in local neighborhoods, determined by the extent of lateral connectivity. A simple Hebbian rule is used to update synaptic weights.

The basic operation of the network is as follows. Let \mathbf{X}_1 denote the input vector from the on-center LGN channel and \mathbf{X}_2 the input vector from the off-center LGN channel to a cortical unit. Each cortical unit receives projections from only a restricted portion of the LGN.

We extend our earlier work [18] by using a mixture of two types of statistics for the input, X , such that

$$X = U + V \quad (1)$$

where U and V denote low frequency and high frequency channels respectively.

Let w_{1Uij} denote a synaptic weight, which represents the strength of the connection between the i^{th} on-center LGN unit for the low frequency input channel U , and the j^{th} unit in the cortex. The subscript numeral "1" represents the on-center LGN unit. Similarly, w_{1Vij} denotes the weight between the i^{th} on-center LGN unit for the high frequency input channel V , and the j^{th} unit in the cortex. Let the subscript numeral "2" denote the off-center LGN units. In a similar fashion, w_{2Uij} is the weight between the i^{th} off-center LGN unit for the

low frequency input channel U , and the j^{th} unit in the cortex, and w_{2Vij} is the weight for the high frequency input channel V .

The output y_j of the j^{th} cortical unit is given by

$$y_j = \sum_{i \in \mathbf{L}_j} w_{1Uij} X_{1Ui} + \sum_{i \in \mathbf{L}_j} w_{1Vij} X_{1Vi} + \sum_{i \in \mathbf{L}_j} w_{2Uij} X_{2Ui} + \sum_{i \in \mathbf{L}_j} w_{2Vij} X_{2Vi} \quad (2)$$

Here the cortical unit combines the responses from the two input frequency channels processed via two LGN channels, and \mathbf{L}_j is the neighborhood of LGN units that project to this j^{th} cortical unit.

The next step is for each cortical unit to determine whether it is a winner within its local neighborhood. Let \mathbf{N}_j denote the local cortical neighborhood of the j^{th} cortical unit (which excludes the j^{th} unit). Let m index the cortical units within \mathbf{N}_j . Thus, unit j is a local winner if

$$\forall m \in \mathbf{N}_j, \quad y_j > y_m \quad (3)$$

This is a local computation for a given cortical unit. Once the local winners are determined, their weights are updated to move them closer to the input vector. If cortical unit j is the winner, the update rule is

$$\begin{aligned} w_{1Uij} &\leftarrow w_{1Uij} + \mu(X_{1Ui} - w_{1Uij}) \\ w_{1Vij} &\leftarrow w_{1Vij} + \mu(X_{1Vi} - w_{1Vij}) \end{aligned} \quad (4)$$

where i indexes those input units that are connected to the cortical unit j , and μ is the learning rate. μ is typically set to a small value, so that the weights are incrementally updated over a large set of input presentations. A similar rule is used to learn w_{2Uij} and w_{2Vij} .

In addition, the weights of the cortical units within the neighborhood \mathbf{N}_j , denoted by the index m , are also updated to move closer to their inputs, but with a weighting function $f(d(j, m))$, where $d(j, m)$ is the distance from the unit m to the local winner j . This is given by

$$\begin{aligned} w_{1Uim} &\leftarrow w_{1Uim} + f(d(j, m))\mu [X_{1Ui} - w_{1Uim}] \\ w_{1Vim} &\leftarrow w_{1Vim} + f(d(j, m))\mu [X_{1Vi} - w_{1Vim}] \end{aligned} \quad (5)$$

and similarly for w_{2Uim} and w_{2Vim} .

Finally, the incident weights at each cortical unit are normalized. The cortical dynamics and learning are thus based on a modified Kohonen algorithm.

Typically, the size of the neighborhood \mathbf{N}_j and the learning rate μ are gradually decreased according to a schedule such that the resulting solution is stable. We use a parameterless-SOM technique to automatically implement such a schedule, and further details are presented in [18].

3.2 Network Configuration

We used an input layer consisting of 30x30 retinal units. The size of the cortex was 60x60 units. Images incident on this simulated retina consisted of sinusoidal gratings of random orientation and phase. The high frequency channel

V , consisted of gratings of spatial frequency four times as high as that in the low frequency channel, U . The orientation and phases of each channel were randomized independently of each other. We emphasize that our model uses real input images, as opposed to other work that uses abstract variables that encode orientation, such as in [6].

The LGN was the same size as the retina. A radius of $r = 9$ was used to generate a topographic mapping from the LGN into the cortex. The intra-cortical connectivity was initialized with a radius of $r_{CC} = 15$.

For the weight updates, the function f was chosen to be a Gaussian that tapers to approximately zero at the boundary of the local neighborhood, ie at r_{CC} . The learning rules in section 3.1 were applied to learn the afferent weights.

4 Experimental Results

We wish to measure the correlational structure of the cortical weights. The general form of the equations is as follows, where $\Psi(\mathbf{x})$ represents the field of weight vectors.

$$c(d) = \sigma^{-1} \left\langle \Re \left[(\Psi(\mathbf{x}) - \bar{\Psi})(\Psi^*(\mathbf{x} + \mathbf{x}') - \bar{\Psi}^*) \right] \right\rangle_{|\mathbf{x}|=d} \quad (6)$$

where the brackets signify average over \mathbf{x} and over the ensemble of cortical weights over each cortical location, $\bar{\Psi}$ is the average field, $*$ is the complex conjugate operator, \Re the real part operator, and σ the variance, defined as:

$$\sigma = \left\langle \Re \left[(\Psi(\mathbf{x}) - \bar{\Psi})(\Psi^*(\mathbf{x}) - \bar{\Psi}^*) \right] \right\rangle \quad (7)$$

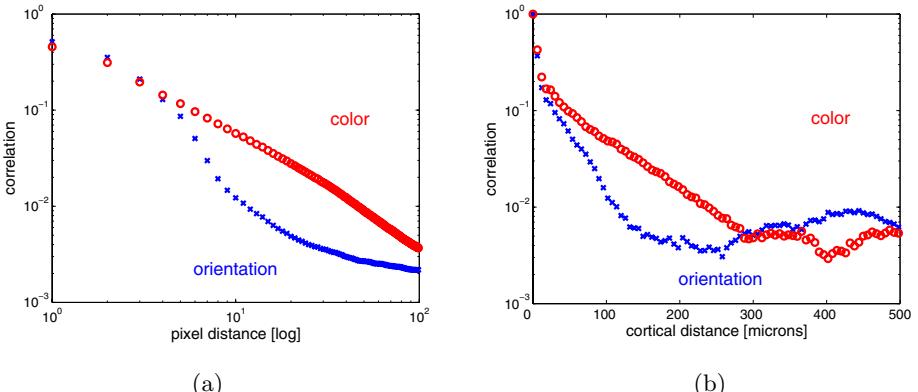


Fig. 3. (a) Spatial auto-correlation of the orientation and color fields in natural images. (b) Spatial auto-correlation of the orientation and color fields in the macaque visual cortex. The fields were obtained by applying a classification approach to the response to color and orientation stimuli [23]. The distance is measured in microns over the cortical surface.

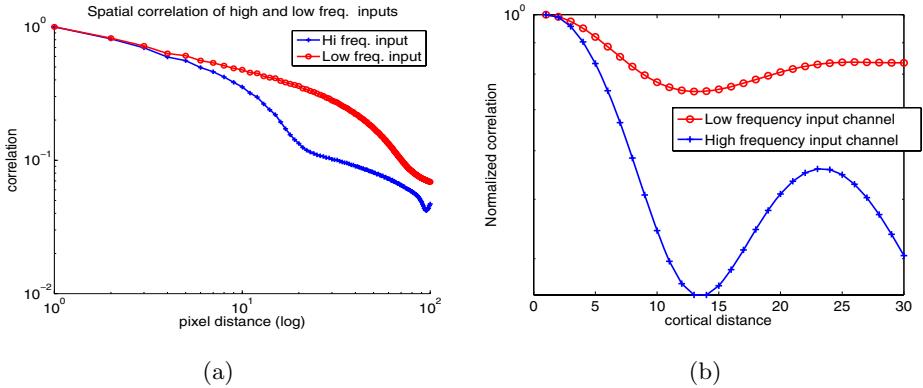


Fig. 4. (a) Spatial correlation of high and low frequency image inputs used for the simulation. (b) Spatial correlation of high and low frequency channels of cortical representations, $c_U(d)$ and $c_V(d)$.

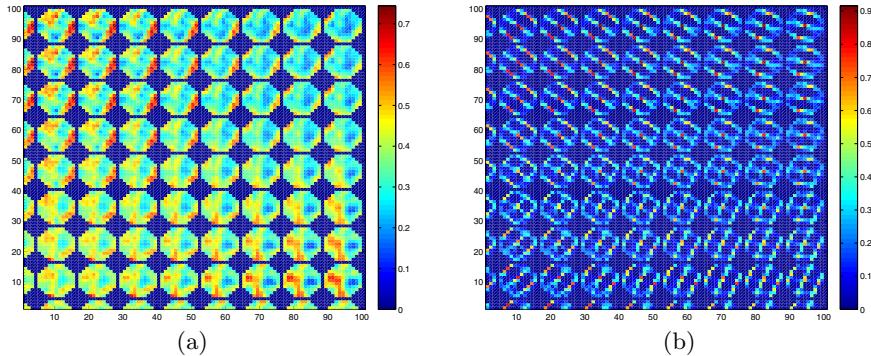


Fig. 5. (a) The cortical weight matrices w_{1U} for the low frequency channel. There are several multi-colored blobs, with each blob representing a cortical unit. Each blob visualizes the rasterized weight matrix that connects the LGN to the corresponding cortical unit. The colormap is indicated by the legend. In order to display the weight matrices clearly in the available space, we show only the central 10×10 portion of the 60×60 cortex. (b) Similarly, we show the cortical weight matrices w_{1V} for the high frequency channel.

where, as in Eq. (6) the average is over \mathbf{x} and the ensemble of cortical weights. The function $c(d)$ is computed for the low frequency cortical channel U and denoted as $c_U(d)$. Similarly, $c_V(d)$ is the correlation for the high frequency cortical channel V .

For the sake of comparison, we first present our previous results from [7] in Figures 3(a) and (b).

For the sake of comparison with Figure 3(a), we show in Figure 4(a) the corresponding spatial correlation of the input image statistics that were used in the simulation.

Figure 4(b) shows the result of applying the computation in Eq. 6 to the two sets of cortical weights, w_U and w_V . This result shows that the *cortical* representation of low frequency inputs is also low frequency, and similarly for high frequency inputs.

Figure 5 shows the receptive fields of the cortical units. Typically, only the preferred input for a given cortical location is shown. However, we display the entire weight matrix at multiple contiguous cortical locations to help understand the correlational structure displayed by the cortical weights.

5 Discussion

We can compare the statistical properties of a macaque visual cortex in Fig. 3(b) with the simulated cortex in Figure 4(b). This comparison is intended to be qualitative, and captures an important aspect of the ecological theory of perception, which is that the cortical representation of the lower frequency stimulus should be spatially more highly correlated than that of the higher frequency stimulus. The color field exhibits lower spatial frequency in the input images as well as in the cortical representation. Similarly, the orientation field exhibits higher spatial frequency.

Kohonen's algorithm does not predict what the correlational structure of cortical organization will be, especially with respect to inputs of different spatial frequencies. In this sense, the question we are addressing, which is the investigation of the similarity of correlation structure in inputs and cortical representation, is non-trivial. Its investigation most likely requires detailed computational simulations, which is a contribution of this paper.

The neural coding implication of our results is that the cortical weights for fast spatially varying inputs, such as orientation should also vary quickly. Whereas, for slower varying input features, such as color, it is sufficient for the cortical representation to also vary slowly.

Our simulations can be made more realistic by using natural images as inputs, which is a direction we are working towards.

The close similarity between the plots shown in Figures 3(a) and Fig. 4(a), and Fig. 3(b) and 4(b) provide confirmatory evidence for the ecological theory of perception as manifested in the representation of visual features in cortical area V1.

6 Conclusions

We presented a novel simulation of cortical organization with respect to inputs of different spatial frequencies. We show that the correlational structure of a simulated visual cortex is similar to that of a real macaque cortex in that the cortical representation of low frequency inputs is also of low frequency, and the representation of high-frequency inputs is of high frequency. Furthermore, the correlation structure of cortical representations matches that found in natural images.

Our results confirm the ecological theory of perception, and show that cortical representations match the statistics of the environment. This finding should spur further neuroscientific experimentation to validate our results. We are in the process of embarking on such a study.

Acknowledgements

We are grateful to Dr. Ehud Kaplan and Dr. Youping Xiao at the Mt. Sinai School of Medicine, New York, for useful discussions on the cortical representation of visual stimuli.

References

1. Atick, J.J., Redlich, A.N.: What does the retina know about natural scenes? *Neural Comp.* 4, 196–210 (1992)
2. Bednar, J.A.: Learning to see: Genetic and environmental influences on visual development. PhD thesis AI-TR-02-294, Dept. of Computer Sciences, Univ. of Texas, Austin (2002)
3. Bell, A.J., Sejnowski, T.J.: The independent components of natural scenes are edge filters. *Vision Res.* 37, 3327–3338 (1997)
4. Berglund, E., Sitte, J.: The parameterless self-organizing map algorithm. *IEEE Trans. Neural Networks* 17(3), 305–316 (2006)
5. Betsch, B.Y., et al.: The world from a cat's perspective - statistics of natural videos. *Biological Cybernetics* 90(1), 41–50 (2004)
6. Carreira-Perpinan, M.A., et al.: A computational model for the development of multiple maps in primary visual cortex. *Cerebral Cortex* 15, 1222–1233 (2005)
7. Cecchi, G.A., et al.: Statistics of natural images and the cortical representation of color. In: Proc. SPIE, vol. 6806 (2008)
8. Erwin, E., Obermayer, K., Schulten, K.: Models of orientation and ocular dominance columns in the visual cortex: A critical comparison. *Neural Computation* 7(3), 425–468 (1995)
9. Field, D.J.: Relations between the statistics of natural images and the response profiles of cortical cells. *J. Opt. Soc. Am. A* 4, 2379–2394 (1987)
10. Grinvald, A., et al.: Functional architecture of cortex revealed by optical imaging of intrinsic signals. *Nature* 324, 361–364 (1986)
11. Hubel, D.H., Wiesel, T.N., Levay, S.: Plasticity of ocular dominance columns in monkey striate cortex. *Phil. Trans. R. Soc. Lond. B* 278, 377–409 (1977)
12. Kohonen, T.: The self-organizing map. *Proceedings of the IEEE* 78(9), 1464–1480 (1990)
13. Li, B., Peterson, M.R., Freeman, R.D.: Oblique effect: A neural basis in the visual cortex. *J/ Neurophysiology* 90, 204–217 (2003)
14. Li, Z.: A neural model of contour integration in the primary visual cortex. *Neural Computation* 10(4), 903–940 (1998)
15. Miller, K.D.: A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between on- and off-center inputs. *J. Neurosci.* 14, 409–441 (1994)
16. Northoff, G.: *Philosophy of the Brain: The Brain Problem*. John Benjamins Publishing, Amsterdam (2003)

17. Obermayer, K., Sejnowski, T.J. (eds.): Self-Organizing Map Formation. MIT Press, Boston (2001)
18. Rao, A.R., et al.: Emergence of topographic cortical maps in a parameterless local competition network. In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.) ISNN 2007. LNCS, vol. 4492, pp. 552–561. Springer, Heidelberg (2007)
19. Rao, A.R., et al.: A model of the formation of a self-organized cortical representation of color. In: Proc. SPIE, vol. 5666, pp. 17–26 (2005)
20. Ruderman, D.: Origins of scaling in natural images. Vision Res. 37, 3385–3398 (1997)
21. Ruderman, D., Bialek, W.: Statistics of natural images: scaling in the woods. Phys. Rev. Lett. 73, 814–817 (1994)
22. Sigman, M., et al.: On a common circle: natural scenes and gestalt rules. Proc. Natl. Acad. Sci. USA 98, 1935–1940 (2001)
23. Xiao, Y., et al.: Improved mapping of information distribution across the cortical surface with the support vector machine. Neural Networks 21(2-3), 341–348 (2008)

Height Defuzzification Method on L^∞ Space*

Takashi Mitsuishi¹ and Yasunari Shidama²

¹ University of Marketing and Distribution Sciences, Kobe 651-2188, Japan

Takashi.MITSUISHI@red.umsd.ac.jp

² Shinshu University, Nagano 380-8553, Japan

Abstract. The mathematical framework for studying of a fuzzy approximate reasoning is presented in this paper. One of the defuzzification methods besides the center of gravity method which is the best well known defuzzification method are described. The continuity of the defuzzification methods and its application to a fuzzy feedback control are discussed.

Keywords: Approximate reasoning, Fuzzy control, Optimization, Functional analysis.

1 Introduction

Mamdani has applied the fuzzy theory to the field of control theory as intelligent control [1]. Fuzzy logic control is one of the ways to represent numerically the control given by human language and sensitivity, and it has been applied in various practical control plants. Since 1990's, like the theory of classical control and modern control, many systematized mathematical considerations have been discussed [4]–[5]. In practical use, fuzzy membership functions (fuzzy sets), which represent input and output states in optimal control system, are decided on the basis of the experience of experts in each peculiar plant before. Therefore some acquisition methods of fuzzy inference rules by a neural network and a genetic algorithm have been proposed [7] [8].

The authors have been studying to establish the automatic and computational determination of fuzzy membership functions, which give optimal controls in fuzzy control system [9] [10]. The authors also have been studying to find algorithms that compute optimal solutions. The authors consider fuzzy optimal control problems as problems of finding the minimum (maximum) value of the cost (benefit) function with feedback law constructed by Mamdani method, product-sum-gravity method, and Nakamori method [11]–[13]. These approximate reasoning methods adopt the center of gravity method, and calculate defuzzified value of inference result represented by fuzzy set. This defuzzification method is most widely used. The resulting behavior of fuzzy approximate reasoning using any of these defuzzification methods will be discussed in the following section. The author's study covers the height defuzzification method [14] [15].

* The paper was supported in part by Grant-in-Aid for Young Scientists (B) #19700225 from Japan Society for the Promotion of Science (JSPS).

Since this method does not synthesize the fuzzy set (membership function), this is better than the center of gravity method in respect of high-speed computing.

In this study, two kinds of continuity of defuzzification are discussed. One is Lipschitz continuity on the space of premise valuable. The other is continuity as functional on the set of membership functions. By the continuity as functional and the compactness of the set of membership functions, the existence of an optimal feedback control law in a nonlinear fuzzy feedback control system, in which the feedback laws are determined by IF-THEN type fuzzy rules, are shown. Then it is crucial to investigate the convergence of feedback laws constructed by fuzzy approximate reasoning method and the convergence of solutions of the nonlinear state equation in the fuzzy control system.

2 Fuzzy Feedback Control

\mathbb{R}^n denotes the n -dimensional Euclidean space with the usual norm $\|\cdot\|$. Let $f(y, v) : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$ be a (nonlinear) vector valued function which is Lipschitz continuous. In addition, assume that there exists a constant $M_f > 0$ such that

$$\|f(y, v)\| \leq M_f (\|y\| + |v| + 1) \quad (1)$$

for all $(y, v) \in \mathbb{R}^n \times \mathbb{R}$.

Consider a system given by the following state equation:

$$\dot{x}(t) = f(x(t), u(t)), \quad (2)$$

where $x(t)$ is the state and the control input $u(t)$ of the system is given by the state feedback

$$u(t) = \rho(x(t)).$$

Assume that the controllability is guaranteed in this system. For a sufficiently large $r > 0$,

$$B_r = \{x \in \mathbb{R}^n : \|x\| \leq r\}$$

denotes a bounded set containing all possible initial states x_0 of the system. Let T be a sufficiently large final time. Then, we have

Proposition 1. [1] Let $\rho : \mathbb{R}^n \rightarrow \mathbb{R}$ be a Lipschitz continuous function and $x_0 \in B_r$. Then, the state equation

$$\dot{x}(t) = f(x(t), \rho(x(t))) \quad (3)$$

has a unique solution $x(t, x_0, \rho)$ on $[0, T]$ with the initial condition $x(0) = x_0$ such that the mapping $(t, x_0) \in [0, T] \times B_r \mapsto x(t, x_0, \rho)$ is continuous.

For any $r_2 > 0$, put

$$\Phi = \{\rho : \mathbb{R}^n \rightarrow \mathbb{R} : \text{Lipschitz continuous, } \sup_{u \in \mathbb{R}^n} |\rho(u)| \leq r_2\}. \quad (4)$$

Then, the following (a) and (b) hold.

(a) For any $t \in [0, T]$, $x_0 \in B_r$ and $\rho \in \Phi$,

$$\|x(t, x_0, \rho)\| \leq r_1, \quad (5)$$

where

$$r_1 = e^{M_f T} r + (e^{M_f T} - 1)(r_2 + 1). \quad (6)$$

(b) Let $\rho_1, \rho_2 \in \Phi$. Then, for any $t \in [0, T]$ and $x_0 \in B_r$,

$$\|x(t, x_0, \rho_1) - x(t, x_0, \rho_2)\| \leq \frac{e^{L_f(1+L_{\rho_1})t} - 1}{1 + L_{\rho_1}} \sup_{u \in [-r_1, r_1]^n} |\rho_1(u) - \rho_2(u)|, \quad (7)$$

where L_f and L_{ρ_1} are the Lipschitz constants of f and ρ_1 .

3 Fuzzy Rules and Fuzzy Sets

Assume the feedback law ρ consists of the following m IF-THEN type fuzzy control rules.

RULE 1: *IF x_1 is A_{11} and ... and x_n is A_{1n}*

THEN y is B_1

⋮

RULE i : *IF x_1 is A_{i1} and ... and x_n is A_{in}*

THEN y is B_i

⋮

RULE m : *IF x_1 is A_{m1} and ... and x_n is A_{mn}*

THEN y is B_m (8)

Here, m is the number of fuzzy production rules, and n is the number of premise variables $x_j \in [-r_1, r_1]$. $y \in [-r_2, r_2]$ is consequence variable. The constants r_1 and r_2 are decided by (4) and (6) in the previous section. Let $\mu_{A_{ij}}$ and μ_{B_i} be membership functions of the fuzzy set A_{ij} and B_i , respectively.

Let $C[-r_1, r_1]$ be the Banach space of all continuous real functions on $[-r_1, r_1]$ with the norm $\|\mu\| = \max_{x \in [-r_1, r_1]} |\mu(x)|$. Denote by $L^1[-r_2, r_2]$ the Banach space of

all Lebesgue measurable real functions μ on $[-r_2, r_2]$ such that $\int_{-r_2}^{r_2} |\mu(x)| dx < \infty$.

We also denote by $L^\infty[-r_2, r_2]$ the Banach space of all Lebesgue measurable, essentially bounded real functions on $[-r_2, r_2]$. Let $\Delta_{ij} > 0$ ($1 \leq i \leq m$; $1 \leq j \leq n$). We consider the following two sets of fuzzy membership functions.

$$F_{\Delta_{ij}} = \{\mu \in C[-r_1, r_1] : 0 \leq \mu(x) \leq 1 \text{ for } \forall x \in [-r_1, r_1],$$

$$|\mu(x) - \mu(x')| \leq \Delta_{ij}|x - x'| \text{ for } \forall x, x' \in [-r_1, r_1]\} \quad (9)$$

and

$$G = \{\mu \in L^\infty[-r_2, r_2] : 0 \leq \mu(x) \leq 1 \text{ a.e. } x \in [-r_2, r_2]\}. \quad (10)$$

The set $F_{\Delta_{ij}}$, which is more restrictive than G , contains triangular, trapezoidal and bell-shaped fuzzy membership functions with gradients less than positive value Δ_{ij} . Consequently, if $\Delta_{ij} > 0$ is taken large enough, $F_{\Delta_{ij}}$ contains almost all fuzzy membership functions which are used in practical applications. In section 8, we shall assume that the fuzzy membership functions $\mu_{A_{ij}}$ in “IF” parts of the rules (8) belong to the set $F_{\Delta_{ij}}$. On the other hand, we shall also assume that the fuzzy membership functions μ_{B_i} in “THEN” parts of the rules (8) belong to the set of discontinuous functions G [6].

In the following, we endow the space $F_{\Delta_{ij}}$ with the norm topology on $C[-r_1, r_1]$ and endow the space G with the weak topology $\sigma(L^\infty, L^1)$ on $L^\infty[-r_2, r_2]$.

Put

$$\mathcal{F} = \prod_{i=1}^m \left\{ \prod_{j=1}^n F_{\Delta_{ij}} \right\} \times G^m, \quad (11)$$

where G^m denotes the m times Cartesian product of G . Then, every element $(\mathcal{A}, \mathcal{B})$ of \mathcal{F} is a fuzzy controller given by the IF-THEN type fuzzy control rules (8).

For simplicity, we write “IF” and “THEN” parts in the rules by the following notation: $\mathcal{A}_i = (\mu_{A_{i1}}, \mu_{A_{i2}}, \dots, \mu_{A_{in}})$ ($i = 1, 2, \dots, m$), $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_m)$ and $\mathcal{B} = (\mu_{B_1}, \mu_{B_2}, \dots, \mu_{B_m})$.

The membership function usually used in the fuzzy control and defined on the real number space is triangular, trapezoidal (π -type), bell-shaped, Z-type, and S-type, etc. If closed interval $[-r_1, r_1]$ and $[-r_2, r_2]$ are taken sufficiently large in a practical use, almost all membership functions are included in the set defined by (9) and (10).

Then, the IF-THEN type fuzzy control rules above is called a fuzzy controller, and is denoted by $(\mathcal{A}, \mathcal{B})$. In the rules, the tuple of premise variable $x = x_1, x_2, \dots, x_n$ is called an input information given to the fuzzy controller $(\mathcal{A}, \mathcal{B})$, and y is called an control variable.

4 Approximate Reasoning

Let $(\mathcal{A}, \mathcal{B})$ be a fuzzy controller given by the IF-THEN type fuzzy control rules above. We say that the system (3) is a *fuzzy feedback system* if the control function $u(t)$ is given by the state feedback $u(t) = \rho_{\mathcal{AB}}(x(t))$, where $\rho_{\mathcal{AB}}(x(t))$ is the amount of operation from the fuzzy controller $(\mathcal{A}, \mathcal{B})$ for the input information $x(t)$. Mamdani method is widely used in fuzzy controls because of its simplicity and comprehensibility. Since min-max gravity method uses minimum and maximum operations, because of their nonlinearity, the value of the agreement degree and the gravity might not change smoothly. In addition, it is pointed out that this method is not necessarily appropriate to express the human intuition. Then, Mizumoto proposed the product-sum-gravity method by replacing minimum with product and maximum with summation [16]. In the following, the

approximate reasoning method with height defuzzification method is introduced. If given the input information $x^* = (x_1^*, x_2^*, \dots, x_n^*)$ to the fuzzy controller $(\mathcal{A}, \mathcal{B})$, the procedure for inference is summarized as follows:

Procedure 1. The strength of each rule is calculated by

$$\begin{aligned}\alpha_{m\mathcal{A}_i}(x^*) &= \bigwedge_{j=1}^n \mu_{A_{ij}}(x_j^*), \\ \alpha_{p\mathcal{A}_i}(x^*) &= \prod_{j=1}^n \mu_{A_{ij}}(x_j^*) \quad (i = 1, 2, \dots, m).\end{aligned}$$

Procedure 2. The defuzzified value y_i^* of the fuzzy set in consequent part of i -th rule is given by

$$y_i^* = \frac{\int_{-r_2}^{r_2} y \mu_{B_i}(y) dy}{\int_{-r_2}^{r_2} \mu_{B_i}(y) dy} \quad (i = 1, 2, \dots, m).$$

Procedure 3. Defuzzification stage.

$$\rho_{\mathcal{AB}}(x^*) = \frac{\sum_{i=1}^m y_i^* \alpha_{\mathcal{A}_i}(x^*)}{\sum_{i=1}^m \alpha_{\mathcal{A}_i}(x^*)}$$

where $\alpha_{\mathcal{A}_i}$ is $\alpha_{m\mathcal{A}_i}$ or $\alpha_{p\mathcal{A}_i}$.

Since these calculations are depend on the membership functions, the subscripts A_{ij} , \mathcal{A}_i , \mathcal{A} , B_i , and \mathcal{B} are put on α_m , α_p , and ρ .

5 Admissible Fuzzy Controller

The reasoning calculation and defuzzification are denoted as composite function through the inference procedure from 1 to 3 on the set of premise valuable $[-r_1, r_1]^n$, and depend on the set of membership function. To avoid making the denominator of the expression in the procedure 2 and defuzzification stage equal to 0, for any $\delta > 0$, and $\varepsilon > 0$, we consider the subset

$$\begin{aligned}\mathcal{F}_{\delta, \varepsilon} &= \left\{ (\mathcal{A}, \mathcal{B}) \in \mathcal{F} : \forall x \in [-r_1, r_1]^n, \sum_{i=1}^m \alpha_{\mathcal{A}_i}(x) \geq \varepsilon, \right. \\ &\quad \left. \forall i = 1, 2, \dots, m, \int_{-r_2}^{r_2} \mu_{B_i}(y) dy \geq \delta \right\},\end{aligned}\tag{12}$$

where $\alpha_{\mathcal{A}_i}$ is $\alpha_{m\mathcal{A}_i}$ or $\alpha_{p\mathcal{A}_i}$. This is a slight modification of \mathcal{F} by (11). If δ and ε are taken small enough, it is possible to consider $\mathcal{F} = \mathcal{F}_{\delta, \varepsilon}$ for practical applications.

6 Continuity of Defuzzification as Functional

In this section, the continuity of approximate reasoning as functional on the set of membership functions $\mathcal{F}_{\delta,\varepsilon}$ is shown. It is already shown that the calculations in the procedure 1 are continuous even if minimum operation or product operation [11] [12]. That is, if for each $i = 1, 2, \dots, m$,

$$\mu_{A_{ij}^k} \rightarrow \mu_{A_{ij}} \quad (i = 1, 2, \dots, m; j = 1, 2, \dots, n)$$

for $k \rightarrow \infty$ implies

$$\|\alpha_{m\mathcal{A}_i^k} - \alpha_{m\mathcal{A}_i}\|_\infty = \sup_{x \in [-r_1, r_1]^n} |\alpha_{m\mathcal{A}_i^k}(x) - \alpha_{m\mathcal{A}_i}(x)| \rightarrow 0$$

and

$$\|\alpha_{p\mathcal{A}_i^k} - \alpha_{p\mathcal{A}_i}\|_\infty = \sup_{x \in [-r_1, r_1]^n} |\alpha_{p\mathcal{A}_i^k}(x) - \alpha_{p\mathcal{A}_i}(x)| \rightarrow 0.$$

Assume that a sequence $(\mathcal{A}^k, \mathcal{B}^k) \subset \mathcal{F}_{\delta,\varepsilon}$ converges to $(\mathcal{A}, \mathcal{B})$ for the product topology if and only if, for each $i = 1, 2, \dots, m$,

$$\|\alpha_{\mathcal{A}_i^k} - \alpha_{\mathcal{A}_i}\|_\infty \rightarrow 0$$

and

$$\mu_{B_i^k} \rightarrow \mu_{B_i} \text{ for the weak topology } \sigma(L^\infty, L^1), \quad (13)$$

where $\alpha_{\mathcal{A}_i}$ is $\alpha_{m\mathcal{A}_i}$ or $\alpha_{p\mathcal{A}_i}$.

Noting that for all $i = 1, 2, \dots, m$, $\int_{-r_2}^{r_2} \mu_{B_i}(y) dy \geq \delta$ by (12) and the definition of membership function, we have

$$\begin{aligned} |y_i^{*k} - y_i^*| &= \left| \frac{\int_{-r_2}^{r_2} y \mu_{B_i^k}(y) dy}{\int_{-r_2}^{r_2} \mu_{B_i^k}(y) dy} - \frac{\int_{-r_2}^{r_2} y \mu_{B_i}(y) dy}{\int_{-r_2}^{r_2} \mu_{B_i}(y) dy} \right| \\ &\leq \frac{r_2}{\delta^2} \left(2 \left| \int_{-r_2}^{r_2} y \mu_{B_i^k}(y) dy - \int_{-r_2}^{r_2} y \mu_{B_i}(y) dy \right| + r_2 \left| \int_{-r_2}^{r_2} \mu_{B_i^k}(y) dy - \int_{-r_2}^{r_2} \mu_{B_i}(y) dy \right| \right). \end{aligned} \quad (14)$$

Then the defuzzified value y_i^* is continuous on $\mathcal{F}_{\delta,\varepsilon}$ by (13). The following inequality is obtained:

$$\begin{aligned} |\rho_{\mathcal{A}^k \mathcal{B}^k}(x) - \rho_{\mathcal{A} \mathcal{B}}(x)| &= \left| \frac{\sum_{i=1}^m \alpha_{\mathcal{A}_i^k}(x) y_i^{*k}}{\sum_{i=1}^m \alpha_{\mathcal{A}_i^k}(x)} - \frac{\sum_{i=1}^m \alpha_{\mathcal{A}_i}(x) y_i^*}{\sum_{i=1}^m \alpha_{\mathcal{A}_i}(x)} \right| \\ &\leq \frac{m}{\varepsilon^2} \sum_{i=1}^m |y_i^{*k} - y_i^*| + \frac{2mr_2}{\varepsilon^2} \sum_{i=1}^m |\alpha_{\mathcal{A}_i^k}(x) - \alpha_{\mathcal{A}_i}(x)|. \end{aligned}$$

It is easy to lead that the functional ρ is continuous on $\mathcal{F}_{\delta,\varepsilon}$ from above inequality (14).

7 Lipschitz Continuity of Defuzzification

In this section, Lipschitz continuity of the defuzzification as the composite function on the premise variable $[-r_1, r_1]$ is shown. Lipschitz condition is applied to the existence of unique solution of the state equation (2).

7.1 Lipschitz Continuity of Height Method

For all $i = 1, 2, \dots, m$, the following mappings $\alpha_{m\mathcal{A}_i}$ and $\alpha_{p\mathcal{A}_i}$ are Lipschitz continuous on the space of premise variables $[-r_1, r_1]^n$ [11] [12]. That is, for $x, x' \in [-r_1, r_1]^n$

$$|\alpha_{\mathcal{A}_i}(x) - \alpha_{\mathcal{A}_i}(x')| \leq \Delta_{\alpha_i} \|x - x'\|,$$

where Δ_{α_i} is Lipschitz constant of $\alpha_{\mathcal{A}_i}$ ($i = 1, 2, \dots, m$). Since $|\alpha_{\mathcal{A}_i}(x)| \leq 1$ for $\forall x \in [-r_1, r_1]^n$,

$$\varepsilon \leq \sum_{i=1}^m \alpha_{\mathcal{A}_i}(x) \leq m.$$

Then we have, similarly

$$|\rho_{\mathcal{AB}}(x) - \rho_{\mathcal{AB}}(x')| \leq \frac{2mr_2}{\varepsilon^2} \sum_{i=1}^m \Delta_{\alpha_i} \|x - x'\|.$$

This inequality shows Lipschitz continuity of height method.

7.2 Existence of Unique Solution of the State Equation

It is easily seen that every bounded Lipschitz function $\rho : [-r_1, r_1]^n \rightarrow \mathbb{R}$ can be extended to a bounded Lipschitz function $\tilde{\rho}$ on \mathbb{R}^n without increasing its Lipschitz constant and bound. In fact, define $\tilde{\rho} : \mathbb{R}^n \rightarrow \mathbb{R}$ by

$$\tilde{\rho}(x) = \tilde{\rho}(x_1, \dots, x_n) = \begin{cases} \rho(x_1, \dots, x_n), & \text{if } x \in [-r_1, r_1]^n \\ \rho(\eta(x_1)r_1, \dots, \eta(x_n)r_1), & \text{if } x \notin [-r_1, r_1]^n, \end{cases}$$

where

$$\eta(u) = \begin{cases} 1, & \text{if } u > r_1 \\ -1, & \text{if } u < -r_1. \end{cases}$$

Let $(\mathcal{A}, \mathcal{B}) \in \mathcal{F}_{\delta, \varepsilon}$. Then it follows from Lipschitz continuity in the previous section and the fact above that the extension $\tilde{\rho}_{\mathcal{AB}}$ of $\rho_{\mathcal{AB}}$ is Lipschitz continuous on \mathbb{R}^n with the same Lipschitz constant of $\rho_{\mathcal{AB}}$ and satisfies $\sup_{u \in \mathbb{R}^n} |\tilde{\rho}_{\mathcal{AB}}(u)| \leq r_2$.

Therefore, by proposition the state equation (3) for the feedback law $\tilde{\rho}_{\mathcal{AB}}$ has a unique solution $x(t, x_0, \tilde{\rho}_{\mathcal{AB}})$ with the initial condition $x(0) = x_0$ [18]. Though the extension $\tilde{\rho}_{\mathcal{AB}}$ of $\rho_{\mathcal{AB}}$ is not unique in general, the solution $x(t, x_0, \tilde{\rho}_{\mathcal{AB}})$ is uniquely determined by $\rho_{\mathcal{AB}}$ using the inequality (7) of (b) of proposition. Consequently, in the following the extension $\tilde{\rho}_{\mathcal{AB}}$ is written as $\rho_{\mathcal{AB}}$ without confusion.

8 Application for Optimization

In this section, using an idea and framework mentioned in the previous section, the existence of optimal control based on fuzzy rules in the admissible fuzzy controller will be established.

The performance index of this control system for the feedback law ρ in the previous section is evaluated with the following integral performance function:

$$J = \int_{B_r} \int_0^T w(x(t, \zeta, \rho), \rho(x(t, \zeta, \rho))) dt d\zeta. \quad (16)$$

where $w : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$ is a positive continuous function. The following theorem guarantee the existence of a rule set which minimizes the previous function (16).

Theorem 1. *The mapping*

$$\mathcal{F}_{\delta, \varepsilon} \ni (\mathcal{A}, \mathcal{B}) \mapsto \int_{B_r} \int_0^T w(x(t, \zeta, \rho_{\mathcal{AB}}), \rho_{\mathcal{AB}}(x(t, \zeta, \rho_{\mathcal{AB}}))) dt d\zeta$$

has a minimum value on the compact metric space $\mathcal{F}_{\delta, \varepsilon}$ defined by (12).

Proof. It is sufficient to prove that compactness of $\mathcal{F}_{\delta, \varepsilon}$ and the continuity of performance function J on $\mathcal{F}_{\delta, \varepsilon}$ are obtained.

For each $i = 1, 2, \dots, m$, $\mathcal{F}_{\Delta_{ij}}$ is a subset of $C[-r_1, r_1]$ which is the subspace of continuous function on $[-r_1, r_1]$. Then it is compact respect for uniform norm $\|\cdot\|_\infty$ by the Ascoli Arzela's theorem [19]. On the other, G is closed for $\sigma(L^\infty, L^1)$. Then it is a compact set respect for the weak topology [11]. Therefor, by the Tychonoff's theorem, \mathcal{F} is compact respect for the product topology. Assume that $\{(\mathcal{A}^k, \mathcal{B}^k)\} \subset \mathcal{F}_{\delta, \varepsilon} \rightarrow (\mathcal{A}, \mathcal{B}) \in \mathcal{F}$ for $k \rightarrow \infty$. Fix $x \in [-r_1, r_1]^n$. It is easy to show that for $i = 1, 2, \dots, m$

$$\int_{-r_2}^{r_2} \mu_{B_i}(y) dy = \lim_{k \rightarrow \infty} \int_{-r_2}^{r_2} \mu_{B_i^k}(y) dy \geq \delta$$

and

$$\sum_{i=1}^m \alpha_{\mathcal{A}_i}(x) = \lim_{k \rightarrow \infty} \sum_{i=1}^m \alpha_{\mathcal{A}_i^k}(x) \geq \varepsilon,$$

and these inequality imply $(\mathcal{A}, \mathcal{B}) \in \mathcal{F}_{\delta, \varepsilon}$. Therefore $\mathcal{F}_{\delta, \varepsilon}$ is a closed subset of \mathcal{F} and hence it is compact. Assume that $(\mathcal{A}^k, \mathcal{B}^k) \rightarrow (\mathcal{A}, \mathcal{B})$ in $\mathcal{F}_{\delta, \varepsilon}$ and fix $(t, \zeta) \in [0, T] \times B_r$. Then it follows from the section 6 that

$$\lim_{k \rightarrow \infty} \sup_{x \in [-r_1, r_1]^n} |\rho_{\mathcal{A}^k \mathcal{B}^k}(x) - \rho_{\mathcal{A} \mathcal{B}}(x)| = 0. \quad (17)$$

Hence, by (b) of proposition, we have

$$\lim_{k \rightarrow \infty} \|x(t, \zeta, \rho_{\mathcal{A}^k \mathcal{B}^k}) - x(t, \zeta, \rho_{\mathcal{A} \mathcal{B}})\| = 0. \quad (18)$$

Further, it follows from (17), (18) and (a) of proposition that

$$\lim_{k \rightarrow \infty} \rho_{\mathcal{A}^k \mathcal{B}^k}(x(t, \zeta, \rho_{\mathcal{A}^k \mathcal{B}^k})) = \rho_{\mathcal{AB}}(x(t, \zeta, \rho_{\mathcal{AB}})). \quad (19)$$

Noting that $w : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$ is positive and continuous, it follows from (18), (19) and the Lebesgue's dominated convergence theorem [20] that the mapping

$$(\mathcal{A}, \mathcal{B}) \in \mathcal{F}_{\delta, \varepsilon} \mapsto \int_{B_r} \int_0^T w(x(t, \zeta, \rho_{\mathcal{AB}}), \rho_{\mathcal{AB}}(x(t, \zeta, \rho_{\mathcal{AB}}))) dt d\zeta$$

is continuous on the compact metric space $\mathcal{F}_{\delta, \varepsilon}$. Thus it has a minimum (maximum) value on $\mathcal{F}_{\delta, \varepsilon}$, and the proof is complete.

9 Conclusion

In this paper, we analyzed the continuity of the defuzzification and proved that there exists an optimal feedback control law in a nonlinear fuzzy feedback control system, in which the feedback laws are determined by IF-THEN type fuzzy rules. Actually the height method is particular case of the area method. To select the proper defuzzification method, it is necessary to analyze the property of fuzzy approximate reasoning and the simulation of fuzzy control.

It is recognized that in various applications it could be a useful tool in analyzing the convergence of fuzzy control rules modified recursively.

References

1. Mamdani, E.H.: Application of fuzzy algorithms for control of simple dynamic plant. Proc. IEE 121(12), 1585–1588 (1974)
2. Tanaka, K., Sugeno, M.: Stability Analysis of Fuzzy Systems and Construction Procedure for Lyapunov Functions. Transactions of the Japan Society of Mechanical Engineers (C) 58(550), 1766–1772 (1992)
3. Hojo, T., Terano, T., Masui, S.: Fuzzy Feedback Control Rules Based on Optimality. Journal of Japan Society for Fuzzy Theory and Systems 5(5), 1200–1211 (1993)
4. Diamond, P.: Stability and periodicity in fuzzy differential equations. IEEE Trans. Fuzzy Syst. 8(5), 583–590 (2000)
5. Furuhashi, T.: Stability Analysis of Fuzzy Control Systems Based on Symbolic Expression. Journal of Japan Society for Fuzzy Theory and Systems 14(4), 357–365 (2002)
6. Gonda, E., Miyata, H., Ohkita, M.: Self-Tuning of Fuzzy Rules with Different Types of MSFs. Journal of Japan Society for Fuzzy Theory and Intelligent Informatics 16(6), 540–550 (2004)
7. Ishibuchi, H., Nii, M.: Generating Fuzzy Classification Rules from Trained Neural Networks. Journal of Japan Society for Fuzzy Theory and Systems 9(4), 512–524 (1997)
8. Nomura, H., Wakami, N.: A Method to Determine Fuzzy Inference Rules by a Genetic Algorithm. The Transactions of the Institute of Electronics, Information and Communication Engineers (A) J77-A(9), 1241–1249 (1994)

9. Shidama, Y., Yang, Y., Eguchi, M., Yamaura, H.: The compactness of a set of membership functions and its application to fuzzy optimal control. *The Japan Society for Industrial and Applied Mathematics* 6(1), 1–13 (1996)
10. Yang, Y., Wasaki, K., Eguchi, M., Shidama, Y., Kimura, M.: The Compactness of a Set of Fuzzy Membership Function in NBV and Its Application. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences* J82-A(4), 523–529 (1999)
11. Mitsuishi, T., Kawabe, J., Wasaki, K., Shidama, Y.: Optimization of Fuzzy Feedback Control Determined by Product-Sum-Gravity Method. *Journal of Nonlinear and Convex Analysis* 1(2), 201–211 (2000)
12. Mitsuishi, T., Kawabe, J., Wasaki, K., Shidama, Y.: Optimization of Fuzzy Feedback Control in L^∞ Space. In: *Proc. International Conference on Fuzzy Systems (FUZZ-IEEE 2001)*, vol. 2, pp. 896–899 (2001)
13. Mitsuishi, T., Endou, N., Shidama, Y.: Continuity of Nakamori Fuzzy Model and Its Application to Optimal Feedback Control. In: *Proc. IEEE International Conference on Systems, Man and Cybernetics*, pp. 577–581 (2005)
14. Mizumoto, M.: Improvement of fuzzy control (II). In: *Proc. 4th Fuzzy System Symposium*, pp. 91–96 (1988)
15. Terano, T.: *Practical Fuzzy Control Technology*. IEICE, Tokyo (1991)
16. Mizumoto, M.: Improvement of fuzzy control (IV) - Case by product-sum-gravity method. In: *Proc. 6th Fuzzy System Symposium*, pp. 9–13 (1990)
17. Nakamori, Y., Ryoke, M.: Identification of fuzzy prediction models through hyperellipsoidal clustering. *IEEE Transactions on Systems, Man and Cybernetics SMC* 24(8), 1153–1173 (1994)
18. Miller, R.K., Michel, A.N.: *Ordinary Differential Equations*. Academic Press, New York (1982)
19. Riesz, F., Sz.-Nagy, B.: *Functional Analysis*. Dover Publications, New York (1990)
20. Dunford, N., Schwartz, J.T.: *Linear Operators Part I: General Theory*. John Wiley & Sons, New York (1988)

An Additive Reinforcement Learning

Takeshi Mori and Shin Ishii

Kyoto University, Gokasho, Uji, Kyoto, 611-0011, Japan

{tak-mori,ishii}@i.kyoto-u.ac.jp

Abstract. In reinforcement learning, preparing basis functions requires a certain amount of prior knowledge and is in general a difficult task. To overcome this difficulty, an adaptive basis function construction technique has been proposed by Keller et al. recently, but it requires excessive computational cost. We propose an efficient approach to this context, in which the problem of approximating the value function is naturally decomposed into a number of sub-problems, each of which can be solved at small computational cost. Computer experiments show that the cpu-time needed by our method is much smaller than that of the existing method.

Keywords: Reinforcement learning, approximation of value function.

1 Introduction

In usual reinforcement learning (RL) schemes, it is important to estimate the action value function that predicts the expected return starting from each state and action [6]. In many realistic RL problems, however, large state and action spaces make the value function estimation in its original function space impractical, and hence the value function approximation using parametric linear models becomes important; the value function is represented as a linear combination of basis functions whose linear coefficients constitute a certain parameter. The parameter is determined by the conventional RL methods based on sample data, but on the other hand the basis functions must be prepared by a designer in advance. However, the preparation is difficult due to the lack of available prior knowledge. A poor setting of basis functions may increase the potential error of approximating the value function, and a large approximation error will cause failures in the policy's learning [2].

One possible idea to deal with this problem is to construct basis functions adaptively based on available data, e.g., by sequentially generating appropriate basis functions based on temporal difference (TD) error or Bellman residual [4]. However, this study employed an inefficient computational procedure; the basis functions were generated based on the TD error in each step, and all the basis functions were used to approximate the value function, hence for calculating the next TD error, by means of the least-squares optimization in each step. This may result in unnecessarily excessive computational cost, especially when the number of basis functions increases as the learning proceeds.

To overcome this difficulty in dealing with an increasing number of basis functions, we propose a novel approach that requires significantly less computational cost than the existing method. We reformulated the problem of estimating the value function into that of estimating the error in the value function, thereby reducing the required number of basis functions. This is because the error, i.e., the decomposed value function, can be more easily approximated than the value function itself. Thus, our method as a whole decomposes the approximation problem in Markov decision processes (MDPs) into a number of approximation sub-problems that are easily solved with a small number of basis functions, so that the parameters are optimized individually within each sub-problem. Computer experiments show that our method is able to significantly reduce not only the number of basis functions but also the computation time in comparison to the previous method [4].

2 MDPs and Value Function Approximation

We consider finite MDPs, each of which is composed of a discrete time t , a set of states \mathbb{S} , a set of actions \mathbb{A} , a set of transition probabilities \mathbb{P} , and a scalar reward $r \in \mathbb{R}$. At time t , the agent selects an action $a_t \in \mathbb{A}$ according to a stationary policy π at a state $s_t \in \mathbb{S}$, and then it moves to a next state $s_{t+1} \in \mathbb{S}$ and simultaneously receives a reward r_{t+1} . The objective is to find the policy that maximizes the action value function:

$$Q^\pi(s, a) = E_\pi \left[\sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1} | s_t = s, a_t = a \right], \quad (1)$$

where $\gamma \in [0, 1]$ is a discount factor.

One of the approaches to seeking an optimal policy is policy iteration [2]. This is composed of two steps, i.e., a policy evaluation step and a policy improvement step. In the former, the value function Q^π for the current policy π is calculated or approximated. In the latter, the policy π is improved based on the learned value function Q ($\approx Q^\pi$). In this article, we focus on the former. The Bellman equation under π is defined as

$$Q^\pi(s, a) = E_\pi[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a]. \quad (2)$$

For the sake of later convenience, we introduce matrix notations: the value function vector $\mathbf{q}^\pi \in \mathbb{R}^{|\mathbb{S}||\mathbb{A}|}$ whose i -th element is the value at the i -th combination of s and a in $\mathbb{S} \times \mathbb{A}$, the state transition matrix $\mathbf{P} \in [0, 1]^{|\mathbb{S}||\mathbb{A}| \times |\mathbb{S}||\mathbb{A}|}$ whose i, j -th element is the transition probability under policy π from the i -th pair of state and action to the j -th pair, and the reward vector $\mathbf{r} \in \mathbb{R}^{|\mathbb{S}||\mathbb{A}|}$ whose i -th element is the expected reward with respect to the next state conditioned on the i -th pair of state and action under π . The Bellman equation (2) can then be simply represented as $\mathbf{q}^\pi = \mathbf{r} + \gamma \mathbf{P} \mathbf{q}^\pi$.

In the context of dynamic programming (DP), the policy evaluation obtains a series of approximations of the value function \mathbf{q} by iterative applications of the Bellman operator $F(\cdot)$:

$$\mathbf{q}^{k+1} = F(\mathbf{q}^k) \equiv \mathbf{r} + \gamma \mathbf{P} \mathbf{q}^k, \quad (3)$$

where k denotes the iteration step; \mathbf{q}^k converges to \mathbf{q}^π at the limit of $k \rightarrow \infty$ [2]. In usual RL settings, however, the vector \mathbf{r} and the matrix \mathbf{P} are unknown, so the value function is estimated by using a sample trajectory, $\langle s_0, a_0, s_1, r_1, a_1, \dots, s_{T-1}, r_{T-1}, a_{T-1}, s_T, r_T \rangle$.

In realistic RL problems, on the other hand, the state and action spaces are large, hence the value function estimation is difficult. Thus the value function is often approximated by using a parametric linear model, i.e., it is represented as a linear combination of basis functions whose linear coefficients constitute the parameter:

$$Q^\pi(s, a) \approx \sum_{m=1}^M \phi_m(s, a) \theta_m \equiv \phi(s, a)' \boldsymbol{\theta}, \quad (4)$$

where $(\cdot)'$ is the transpose and the basis function vector $\phi(s, a)$ and the parameter $\boldsymbol{\theta}$ are both M -dimensional vectors. Note that the designer of the learning system must prepare the basis functions prior to learning. The parameter $\boldsymbol{\theta}$ is adjusted in the policy evaluation step by using the sample trajectory. One of the learning methods is least-squares TD learning (LSTD) [3], which obtains a closed-form solution for the linearly-approximated value function.

3 Previous Method for Basis Function Construction

Keller et al. [4] proposed an adaptive basis function construction technique, in which appropriate basis functions were generated based on the aggregation method proposed formerly by Bertsekas and Castañon [1]. In that method, adaptively-produced aggregated states were used to speed up the value iteration in DP, even though applications of the original Bellman operator (Eq.(3)) can be very slow. According to the aggregation method, the Bellman operator (Eq.(3)) is replaced by its aggregate form:

$$\mathbf{q}^{k+1} = \mathbf{q}^k + \Psi \mathbf{y}, \quad (5)$$

where $\Psi \in \{0, 1\}^{|\mathcal{S}| \times |\mathcal{A}| \times M}$ is an ‘aggregation matrix’ such that $\Psi_{ij} = 1$ if the i -th state and action pair is assigned to group j or $= 0$ otherwise. \mathbf{y} is the parameter vector for Ψ , which is analytically obtained [1]. $\Psi \mathbf{y}$ becomes the orthogonally projected vector of the difference $\mathbf{q}^\pi - \mathbf{q}^k$ onto the subspace spanned by the sparse aggregation matrix Ψ . By applying the Bellman operator F to Eq.(5),

$$F(\mathbf{q}^{k+1}) = F(\mathbf{q}^k) + \gamma \mathbf{P} \Psi \mathbf{y} \quad (6)$$

is obtained, so that the following holds¹:

$$F(\mathbf{q}^{k+1}) - F(\mathbf{q}^k) = \mathbf{g}_1 + \mathbf{g}_2, \quad (7)$$

¹ This is derived from [1] with some modifications for our purpose.

where

$$\mathbf{g}_1 = (\mathbf{I} - \boldsymbol{\Pi})(F(\mathbf{q}^k) - \mathbf{q}^k) \quad \text{and} \quad \mathbf{g}_2 = \gamma(\mathbf{I} - \boldsymbol{\Pi})\mathbf{P}\boldsymbol{\Psi}\mathbf{y}.$$

Here, $\boldsymbol{\Pi} \equiv \boldsymbol{\Psi}(\boldsymbol{\Psi}'\mathbf{D}\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}'\mathbf{D}$ is an orthogonal projection and \mathbf{D} is a diagonal matrix whose elements denote the steady-state or visiting probabilities of the corresponding state-action pairs. Because it is known experimentally that \mathbf{g}_2 often decreases as \mathbf{g}_1 decreases [1], one can focus only on the minimization of \mathbf{g}_1 . An effective method to minimize \mathbf{g}_1 was proposed in [1] such that all the state-action pairs $\{i | i = 0, \dots, |\mathbb{S}||\mathbb{A}|\}$ were assigned to M -disjoint aggregate groups $\{\mathbb{E}_1, \dots, \mathbb{E}_M\}$, according to the ranking of the Bellman residuals $\{(F(\mathbf{q}^k) - \mathbf{q}^k)_i | i = 0, \dots, |\mathbb{S}||\mathbb{A}|\}$, where the solution achieves the ‘minimax optimal’ against the worst possible choices of \mathbf{P} and \mathbf{q}^k [1].

In Keller et al.’s method, by applying the aggregation method to an RL context, the value function is updated as

$$\mathbf{q}^{k+1} = \boldsymbol{\Phi}_k \boldsymbol{\theta}_k + \boldsymbol{\Psi} \boldsymbol{\theta}_{new} = (\boldsymbol{\Phi}_k \quad \boldsymbol{\Psi}) \begin{pmatrix} \boldsymbol{\theta}_k \\ \boldsymbol{\theta}_{new} \end{pmatrix} \equiv \boldsymbol{\Phi}_{k+1} \boldsymbol{\theta}_{k+1},$$

where $\boldsymbol{\Phi}_k \in \{0, 1\}^{|\mathbb{S}||\mathbb{A}| \times kM}$ is the basis function matrix having the form: $\boldsymbol{\Phi}_k = (\phi_k(1, 1) \ \phi_k(1, 2) \ \cdots \ \phi_k(|\mathbb{S}|, |\mathbb{A}|))'$ where $\phi_k(s, a) \in \{0, 1\}^{kM}$ is a kM -dimensional binary vector for the pair of state s and action a , and M is the number of basis functions produced in each step. In this method, the basis functions are constructed iteratively based on the TD error (note that the Bellman residual is the expectation of the TD error), and the parameters are obtained by LSTD [3] [4]. For an observed sample trajectory, the parameter vector $\boldsymbol{\theta}_{k+1}$ is optimized so as to minimize the cost function:

$$J_{LSTD}(\boldsymbol{\theta}_{k+1}) = \frac{1}{2T} \sum_{t=0}^{T-1} (Q^\pi(s_t, a_t) - \phi_{k+1}(s_t, a_t)' \boldsymbol{\theta}_{k+1})^2, \quad (8)$$

where T is the trajectory length. At each iteration step k , a kM -dimensional vector $\boldsymbol{\theta}_k$ is obtained by the least-squares optimization technique; more concretely [3],

$$\boldsymbol{\theta}_k = \mathbf{A}^{-1} \mathbf{b}_k, \quad (9)$$

where $kM \times kM$ matrix \mathbf{A}_k and kM -dimensional vector \mathbf{b}_k are given by

$$\mathbf{A}_k = \sum_{t=0}^{T-1} \phi_k(s_t)(\phi_k(s_t, a_t) - \phi_k(s_{t+1}, a_{t+1}))' \quad \text{and} \quad (10)$$

$$\mathbf{b}_k = \sum_{t=0}^{T-1} \phi_k(s_t, a_t) r_{t+1}, \quad (11)$$

respectively.

Because the number of parameters increases by M after each iteration, computation of the inverse (Eq. (9), $\mathcal{O}((kM)^3)$) and the sum of the outer products (Eq.(10), $\mathcal{O}(T(kM)^2)$) becomes difficult as k increases.

4 Proposed Method

To overcome the above-mentioned computational difficulty, we propose a novel approach to adaptively constructing basis functions, in which the computational cost is greatly reduced in comparison to the Keller et al.'s method. While in the previous method the value function was at each step approximated by all the parameters, we let newly-added parameters $\boldsymbol{\theta}_{new}$ approximate the error in the approximate value function instead of the value function, and already-learned parameters $\boldsymbol{\theta}_k$ be fixed and embedded in \mathbf{q}^k . In this method, the approximation error is approximated at step $k + 1$ by $\Psi\boldsymbol{\theta}_{new}$ based on least-squares optimization, and the value function is updated as

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Psi\boldsymbol{\theta}_{new}. \quad (12)$$

Note that only the M -dimensional parameter vector $\boldsymbol{\theta}_{new}$ is learned, whereas the remaining $\boldsymbol{\theta}_k$ is fixed. Thus, the value function is additively updated as $\mathbf{q}_k, \mathbf{q}_{k+1}, \mathbf{q}_{k+2}, \dots$, until the approximation error becomes sufficiently small.

Next, we explain the least-squares optimization performed to reduce the error in each step. At step $k + 1$, the cost function of the least-squares optimization is given by

$$J(\boldsymbol{\theta}_{new}) = \frac{1}{2T} \sum_{t=0}^{T-1} (Q^\pi(s_t, a_t) - Q^k(s_t, a_t) - \psi(s_t, a_t)' \boldsymbol{\theta}_{new})^2, \quad (13)$$

where $\psi(s_t, a_t) \in \{0, 1\}^M$ is an M -dimensional binary vector for the pair of state s and action a such that $\Psi = (\psi(1, 1) \ \psi(1, 2) \ \dots \ \psi(|\mathbb{S}|, |\mathbb{A}|))'$. The derivative of the cost function with respect to $\boldsymbol{\theta}_{new}$ is given by

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{new}) &= \frac{1}{2} \sum_{t=0}^{T-1} \psi(s_t, a_t) (Q^\pi(s_t, a_t) - Q^k(s_t, a_t) - \psi(s_t, a_t)' \boldsymbol{\theta}_{new}) \\ &\equiv \frac{1}{2} \sum_{t=0}^{T-1} \psi(s_t, a_t) (U_k^\pi(s_t, a_t) - \psi(s_t, a_t)' \boldsymbol{\theta}_{new}). \end{aligned} \quad (14)$$

The target of the regression problem above, $U_k^\pi(s_t, a_t) \equiv Q^\pi(s_t, a_t) - Q^k(s_t, a_t)$, is replaced by the expectation of the discounted cumulative TD error:

$$U_k^\pi(s, a) = E_\pi \left[\sum_{i=0}^{\infty} \gamma^i \delta_{t+i+1} \middle| s_t = s, a_t = a \right], \quad (15)$$

where δ_{t+1} is the TD error:

$$\delta_{t+1} \equiv r_{t+1} + \gamma Q^k(s_{t+1}, a_{t+1}) - Q^k(s_t, a_t). \quad (16)$$

From its definition, Eq. (15), $U_k^\pi(s, a)$ satisfies

$$U_k^\pi(s, a) = E_\pi[\delta_{t+1} + \gamma U_k^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a], \quad (17)$$

which corresponds to the Bellman equation for $U_k^\pi(s, a)$.

If $U_k^\pi(s, a)$ is known, θ_{new} is obtained as the least-squares (LS) solution of the objective function (13) by setting $\nabla J(\theta_{new})$ as $\mathbf{0}$. Because $Q^\pi(s, a)$ is unknown and hence $U_k^\pi(s, a)$ is unknown, however, we introduce a bootstrapping technique [6] to approximate $U_k^\pi(s, a)$ in Eq.(14) such that the target $U_k^\pi(s_t, a_t)$ is replaced by some other target variable u_t , and then the gradient (14) is replaced by

$$\tilde{\nabla}_{\theta} J(\theta_{new}) \equiv \frac{1}{2} \sum_{t=0}^{T-1} \psi(s_t, a_t)(u_t - \psi(s_t, a_t)' \theta_{new}). \quad (18)$$

If u_t is an unbiased estimator, that is, if $E[u_t] = U_k^\pi(s_t, a_t)$ for each t , then θ_{new} , the solution of $\tilde{\nabla}_{\theta} J(\theta_{new}) = \mathbf{0}$, is guaranteed to converge to the LS solution of the objective function (13) as the number of samples (T) increases. For example, if u_t is set as $u_t = \sum_{i=0}^{\infty} \gamma^i \delta_{t+i+1}$ and the sample trajectory is generated under the policy π starting from the state-action pair $s_t = s$ and $a_t = a$, the target u_t is an unbiased estimator (Monte Carlo estimate) of U_k^π , so that θ_{new} converges to the LS solution. Unfortunately, this Monte Carlo method often introduces large variance into the estimation due to the calculation based on the infinitely long trajectory. To deal with this difficulty, the trajectory is truncated by using the Bellman equation (Eq.(17)) such that u_t is set as $u_t = \delta_{t+1} + \gamma \delta_{t+2} + \cdots + \gamma^{m-1} \delta_{t+m} + \gamma^{m-1} U_k^\pi(s_{t+m}, a_{t+m})$, which is still an unbiased estimator of $U_k^\pi(s_t, a_t)$. Because $U_k^\pi(s_{t+m}, a_{t+m})$ is unknown in this equation, it must be replaced by $\psi(s_{t+m}, a_{t+m})' \theta_{new}$, but $\psi(s_{t+m}, a_{t+m})' \theta_{new}$ is not an unbiased estimator of $U_k^\pi(s_{t+m}, a_{t+m})$, and thus this method does not converge to the LS solution. Such biased replacement is sometimes called “bootstrapping” [6]. Nevertheless, such a bootstrapping method can be quite effective, and other favorable characteristics are available in some special cases [2]. In the extreme case, U_k^π is approximated by one-step trajectory, i.e., u_t is set as $\delta_{t+1} + \gamma \psi(s_{t+1}, a_{t+1})' \theta_{new}$, which is an approximation of the right hand side of the Bellman equation (Eq.(17)). In this article, we only explain this extreme for the sake of simplicity. We therefore re-define an approximate derivative $\tilde{\nabla}_{\theta} J(\theta_{new})$ so that the new parameter θ_{new} is obtained as its zero point:

$$\tilde{\nabla}_{\theta} J(\theta_{new}) \equiv \frac{1}{2} \sum_{t=0}^{T-1} \psi(s_t, a_t)(\delta_{t+1} + \gamma \psi(s_{t+1}, a_{t+1})' \theta_{new} - \psi(s_t, a_t)' \theta_{new}) = \mathbf{0}.$$

The solution is given as a closed form:

$$\theta_{new} = \mathbf{B}^{-1} \mathbf{c}, \quad (19)$$

where

$$\mathbf{B} = \frac{1}{T} \sum_{t=0}^{T-1} \psi(s_t, a_t)(\psi(s_t, a_t) - \gamma \psi(s_{t+1}, a_{t+1}))' \quad \text{and} \quad (20)$$

$$\mathbf{c} = \frac{1}{T} \sum_{t=0}^{T-1} \psi(s_t, a_t) \delta_{t+1}. \quad (21)$$

Note that the newly-produced basis functions are only used for reducing the approximation error of the value function in the previous step. This allows the computational cost of the inverse (Eq. (19)) to be of $\mathcal{O}(M^3)$, which is much smaller than $\mathcal{O}((kM)^3)$ of the previous method. Also, the computational cost of Eq. (20) is of $\mathcal{O}(TM^2)$, which is much smaller than $\mathcal{O}(T(kM)^2)$ of the previous method. Moreover, our method can be seen as a straightforward application of Bertsekas and Castaño's aggregation method shown in Eqs.(5)-(7) to the context of RL, since the error in the value function is projected onto the space Ψ in both our method and that of Bertsekas and Castaño. We call this new method "differential LSTD" (dLSTD).

Pseudo-code of dLSTD

(Sampling phase): Generate a sample trajectory: $(s_0, a_0, s_1, r_1, a_1, s_2, r_2, \dots, a_{T-1}, s_T, r_T)$

(Learning phase): Repeat until the value function converges

1. Calculate the TD error (or Bellman residual): $\delta_0, \dots, \delta_{T-1}$
2. Generate the new basis function matrix Ψ based on $\delta_0, \dots, \delta_{T-1}$
3. Calculate \mathbf{B} and \mathbf{c} in Eqs. (20) and (21)
4. $\theta := \mathbf{B}^{-1}\mathbf{c}$
5. Update the value function: $\mathbf{q} := \mathbf{q} + \Psi\theta$

(Policy update phase): Update policy so that $\pi(s) := \arg \max_a Q(s, a)$ for each state s , and go back to (sampling phase) in the case of on-policy, or (learning phase) in the case of off-policy

5 Computer Experiments

We compared our method with the Keller et al.'s method [4] by using two benchmark problems: the 50-state chain problem [5] and the blackjack problem [6].

5.1 Experimental Settings

50-Chain. In this problem, an agent moves through a single chain consisting of 50 states to maximize the expected return defined as accumulated discount rewards. For more details, see [5].

First, as a sampling phase, we generated a single trajectory whose length was $T = 10,000$ by using a random policy. Next, we performed a learning phase through which we evaluated our method. In each step in the learning phase, all pairs of state s and action a were aligned according to the TD error averaged over all the appearances in the trajectory, and grouped into two aggregate basis functions (i.e., the number of the basis function M was set as 2). The policy was updated when the maximum absolute value of the averaged TD errors became smaller than 10^{-10} . The entire RL algorithm was shown in the previous section.

Blackjack. Here, we explain the blackjack problem. The rule of the game and the reward settings are the same as those in [6]. In our experiment, a single state is represented as three variables: the player’s card value total, the card value total the dealer shows, and the player’s option of counting an ace as a value of 1. A state-action pair is represented as a four-dimensional vector \mathbf{x} , with three entries of \mathbf{x} being the state variables and the fourth being the player’s action (i.e., to hit or to stand). Thus, we consider the value function on \mathbf{x} , i.e., $Q(\mathbf{x})$.

First as a sampling phase, by using a random policy for the player, we performed 10,000 games whose average length (i.e., the number of hits) was about 1.44, then the total number of samples (T) was about 14,400 ($= 10,000 \times 1.44$).

We then conducted a learning phase consisting of several iteration steps enough to make the TD errors small. In each iteration step, to aggregate in the four-dimensional state-action space, we first perform a random projection onto a one-dimensional space, and then aggregate in that space; each sample vector \mathbf{x}_t was linearly projected onto a scalar y_t , $y_t = \mathbf{e}'\mathbf{x}_t$ ($t = 0, \dots, T - 1$), where t is the sample index and \mathbf{e} is a four-dimensional vector randomly selected from $[-1, 1]^4$. Then, all the samples $\{\mathbf{x}_t | t = 0, \dots, T - 1\}$ were separated into N -disjoint subsets $\{\mathbb{C}_1, \dots, \mathbb{C}_N\}$, according to the discretization of the projected samples $\{y_t | t = 0, \dots, T - 1\}$, and all the subsets, $\mathbb{C}_1, \dots, \mathbb{C}_N$, were assigned to aggregate groups, $\mathbb{E}_1, \dots, \mathbb{E}_M$, according to the TD error averaged over the samples belonging to each subset. The corresponding basis function of the sample $\mathbf{x}_t \in \mathbb{C}_i \in \mathbb{E}_j$ is $\psi(\mathbf{x}_t)$ whose j -th element is 1 and the other elements are all 0. In our experiment, we set N as 100 and M as 2; then, we consider two aggregate groups \mathbb{E}_1 and \mathbb{E}_2 , and the corresponding basis functions are $\psi(\mathbf{x}_t) = [1, 0]'$ and $\psi(\mathbf{x}_t) = [0, 1]'$.

At the end of each iteration step, we checked whether to update the policy, so that the policy was updated when the maximum absolute value of the averaged TD errors became smaller than 10^{-10} . Remaining of the algorithm follows the one presented in Section 4.

5.2 Experimental Results

We compared the computation time of our method (dLSTD), that of the Keller et al.’s method (LSTD) [4], and that of the Keller et al.’s method with the random basis function (LSTD-random) in which $\psi(\mathbf{x}) = [1, 0]'$ or $[0, 1]'$ was randomly chosen regardless of the value of \mathbf{x} . We implemented them using matlab 7.7.0(R2008b) on 3.00 GHz Intel(R) Xeon(R). Figure 1 shows the learning curves averaged over 100 learning runs for the chain walk problem, where Fig 1(a) shows the comparison in the policy evaluation and Fig 1(b) in the policy improvement. The vertical axes in Fig 1(a) denote the logarithmic maximum norm² of the TD error averaged over all pairs of state and action. The vertical axes in Fig 1(b) denote the logarithmic weighted L2 norm between \mathbf{q}^π and \mathbf{q}^{π^*} , where \mathbf{q}^{π^*} is the value function vector under the optimal policy π^* . \mathbf{D}^{π^*} is a diagonal matrix

² For an L -dimensional vector $\mathbf{x} = [x_1, x_2, \dots, x_L]',$ the maximum norm is defined as $\|\mathbf{x}\|_\infty \equiv \max\{|x_1|, |x_2|, \dots, |x_L|\}.$

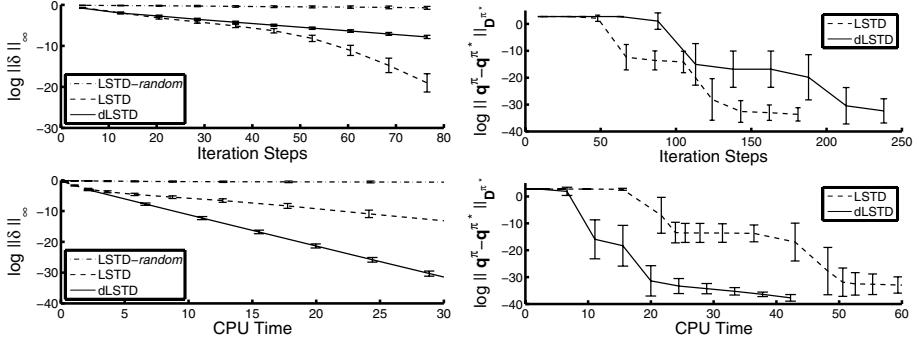


Fig. 1. Chain walk

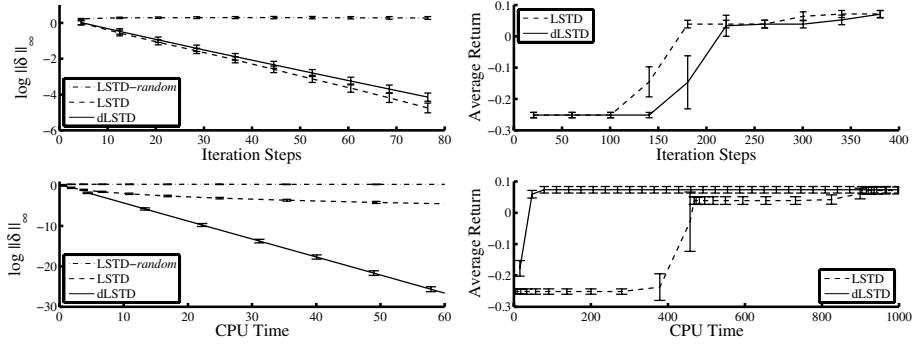


Fig. 2. Blackjack

whose i -th element is the probability of visiting and selecting the i -th pair of s and a under π^* . The horizontal axes in the upper panels of Fig. 1 denote the iteration steps, and those in the lower panels denote the CPU time (sec.).

In Fig. 1, we can see that dLSTD is much faster than LSTD in terms of the CPU time, both for policy evaluation and policy improvement. This is because dLSTD tried to approximate the error in the value function by using a small number (two in this experiment's ($M = 2$)) basis functions. On the other hand, LSTD tried to approximate the value function itself by using a large number of basis functions (i.e., kM where k is the number of iteration steps), hence the computation came to be difficult as the number of iteration steps k increased.

Figure 2 shows the learning curves averaged over 100 learning runs for the blackjack problem. The vertical axes of Fig. 2(b) denote the average return. Similar to Fig. 1, dLSTD was faster than LSTD both in the policy evaluation and the policy improvement. Especially in Fig. 2(b), dLSTD achieved its best policy by 50 sec., which was much smaller (i.e., about 900 sec. lower) than the value at which LSTD showed comparable performance.

In terms of iteration steps, the Keller et al.'s method was better than ours (upper panels of Figs 11 and 12), because it approximates the value function within the full space consisting of all available basis functions, at the sacrifice of computational cost for optimization over the full space.

Throughout our experiments, the performance of LSTD with randomly generated basis functions (LSTD-random) was always worst, showing that there is a high demand for effectiveness in the basis function construction.

6 Conclusion

In this study, we proposed a novel scheme for automatic construction of basis functions, with the absence of large computational cost. In this method, the value function was estimated by sequentially approximating its approximation error by using newly-added basis functions. Our experiments showed that our method is much faster than the existing method [4], while maintaining its approximation performance.

References

1. Bertsekas, D., Castañon, D.: Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control* 34, 589–598 (1989)
2. Bertsekas, D.P., Tsitsiklis, J.N.: *Neuro-Dynamic Programming*. Athena Scientific, Belmont (1996)
3. Bradtko, S.J., Barto, A.G.: Linear least-squares algorithms for temporal difference learning. *Machine Learning* 22(2), 33–57 (1996)
4. Keller, P.W., Mannor, S., Precup, D.: Automatic basis function construction for approximate dynamic programming and reinforcement learning. In: *Proceedings of the Twenty-third International Conference on Machine Learning* (2006)
5. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. *Journal of Machine Learning Research* 4, 1107–1149 (2003)
6. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge (1998)

Neural Spike Suppression by Adaptive Control of an Unknown Steady State

Arūnas Tamaševičius¹, Elena Tamaševičiūtė^{1,2}, Gytis Mykolaitis^{1,3},
Skaidra Bumelienė¹, Raimundas Kirvaitis⁴, and Ruedi Stoop⁵

¹ Plasma Phenomena and Chaos Laboratory, Semiconductor Physics Institute,
A. Goštauto 11, LT-01108 Vilnius, Lithuania
tamasev@pfi.lt, skaidra@pfi.lt
http://www.pfi.lt/index_e

² Department of General Physics and Spectroscopy, Faculty of Physics,
Vilnius University,
Saulėtekio 9, LT-10222 Vilnius, Lithuania
elena.tamaseviciute@ff.vu.lt
<http://www.vu.lt/en/>

³ Department of Physics, Faculty of Fundamental Sciences,
Vilnius Gediminas Technical University,
Saulėtekio 11, LT-10223 Vilnius, Lithuania
gytis@pfi.lt
<http://www.fm.vgtu.lt/en/>

⁴ Department of Electronic Systems, Faculty of Electronics,
Vilnius Gediminas Technical University,
Naugarduko 41, LT-03227 Vilnius, Lithuania
raimundas.kirvaitis@adm.vgtu.lt
<http://www.el.vgtu.lt/en/>

⁵ Institute of Neuroinformatics, University of Zürich (UNIZH) and
Swiss Federal Institute of Technology Zürich (ETHZ),
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
ruedi@ini.phys.ethz.ch
<http://stoop.ini.uzh.ch>

Abstract. A FitzHugh–Nagumo type spiking neuron model equipped with an asymmetric activation function is investigated. An analogue nonlinear electrical circuit imitating the dynamics of the model is proposed. It is demonstrated that a simple first order linear filter coupled to the system can inhibit spiking and stabilize the system on an unstable steady state, the position of which is not required to be known, since the filter operates as an adaptive controller. Analytical, numerical and experimental results are presented.

Keywords: Adaptive control, Nonlinear dynamics, Neuron models.

1 Introduction

The problem of stabilizing unstable steady states (USS) is of great importance in nonlinear dynamics. Classical control methods require as a reference point the

coordinates of the USS. In many practical cases the location of the USS is either unknown or it may slowly vary with time. Therefore adaptive, reference-free methods, automatically locating the USS are preferable.

The simplest adaptive technique for stabilizing USS is based on derivative controller. A perturbation in the form of a derivative dx/dt derived from an observable $x(t)$ does not change the original system, since it vanishes when the variable approaches the steady state. This technique has been applied to stabilize a laser [1], an electrical circuit [2] and an electrochemical reaction [3]. Since the method requires differentiation it is rather sensitive to high frequency noise present in the signal $x(t)$.

Another adaptive method for stabilizing USS employs first order RC filters in the feedback loop [4,5,6,7,8,9,10]. Provided the cut-off frequency of the RC filter is low enough, the voltage across the capacitor of the filter asymptotically approaches the USS and therefore can be used as a reference point for the proportional feedback. The method has been successfully applied to several experimental systems, including electrical circuits [4,5,6] and lasers [7,8]. Recently its modifications have been used to control unstable spirals in mathematical models, such as the Lorenz system [9,10], the van der Pol oscillator [10], and the parametrically excited mechanical pendulum [10].

Neuron models are dynamical systems exhibiting a rich variety of nonlinear phenomena, ranging from excitability, spiking, bursting, chaotic response to external periodic forcing, high-frequency inhibitory effects, to various spatio-temporal patterns in networks of coupled neurons. The FitzHugh–Nagumo (FHN) model [11] is one of the most popular models, due to its simplicity. Besides analytical and numerical studies of the different models, efforts have also concentrated on designing and building analogue electronic neurons. Replacement of biological neurons with electronic devices governed by simple equations, can contribute to a better understanding of the biological effects. A number of nonlinear electrical circuits imitating dynamical behaviour of neurons have been described in the literature. Among them are various modifications of the FHN electronic cells [12,13,14,15], the Hodgkin–Huxley (HH) neuron type [16], and the Liao’s time delayed [17] electronic neurons. Some earlier electronic implementations, e.g. of the Lewis and the Harman models, are collected in Ref. [18].

In the present paper, we describe an extremely simple modification of the FHN model by an asymmetric activation function, to which, for the sake of brevity, we will refer to as the asymmetric FHN (AFHN) model. We derive its experimental electronic analog and demonstrate analytically, numerically and experimentally that the USS can be robustly controlled by a simple first order filter.

2 Mathematical Model

We consider the following set of equations:

$$\dot{x} = ax - f_d(x) - y - \xi, \quad (1)$$

$$\dot{y} = x - by, \quad (2)$$

$$f_d(x) = \begin{cases} d_1(x+1), & x < -1 \\ 0, & -1 \leq x \leq 1 \\ d_2(x-1), & x > 1 \end{cases}. \quad (3)$$

All coefficients in Eqs. (1–3) are positive and constant, except the parameter ξ , which, in general, will be unknown and/or may slowly vary with time. Though the coefficients d_1 and d_2 are somewhat arbitrary (the only requirement is $d_{1,2} > a$), we consider the case of strong asymmetry $d_2 \ll d_1$. Eqs. (1–3) can be treated as asymmetric FHN (AFHN) equations, where the common activation term $x-x^3$ is replaced with an asymmetric activation function $F(x) = ax-f_d(x)$. The function $F(x)$ has essentially different slopes d_1 at negative ($x < -1$) and d_2 at positive ($x > 1$) values of x . For

$$ab < 1, \quad |\xi| < 1/b - a \quad , \quad (4)$$

Eqs.(1–3) lead a single unstable steady state of coordinates

$$x_0 = -\frac{b\xi}{1-ab}, \quad y_0 = -\frac{\xi}{1-ab} \quad , \quad (5)$$

while there are no stable solutions. Thus, conditions (4) yield oscillatory solutions of Eqs. (1–2). Evidently, $|x_0| < 1$ and for $\xi > 0$ the steady state is negative $x_0 < 0, y_0 < 0$. When linearized around the steady state (x_0, y_0) , system (1–2) leads to the characteristic equation

$$\lambda^2 - (a-b)\lambda + 1 - ab = 0, \quad (6)$$

which has two solutions that are independent on ξ :

$$\lambda_{1,2} = \frac{a-b}{2} \pm \sqrt{\frac{(a+b)^2}{4} - 1}. \quad (7)$$

For $a > b$, the real parts of $\lambda_{1,2}$ are positive, confirming that the steady state given by (5) is unstable (either an unstable spiral or an unstable node). When $a > b$ and $a+b > 2$ both solutions are positive and real (no imaginary part). In this case, the steady state is an unstable node.

Now we add to Eqs. (1–2) the third equation describing a stable RC filter and couple it to the AFHN equations:

$$\dot{x} = ax - f_d(x) - y - \xi - k(x-z), \quad (8)$$

$$\dot{y} = x - by, \quad (9)$$

$$\dot{z} = \omega_f(x-z), \quad (10)$$

where k is the coupling coefficient and ω_f is the normalized threshold frequency of the filter. This system has the same steady state solution as the free-running system:

$$x_0 = z_0 = -\frac{b\xi}{1-ab}, \quad y_0 = -\frac{\xi}{1-ab}, \quad (11)$$

which implies that the filter does not influence the position of the steady state (x_0, y_0) of the AFHN system, but can change its stability properties.

The corresponding characteristic equation of the linearized system is

$$\lambda^3 + h_3\lambda^2 + h_2\lambda + h_1 = 0, \quad (12)$$

with

$$h_3 = -a + b + k + \omega_f, \quad h_2 = 1 - ab + bk - (a - b)\omega_f, \quad h_1 = (1 - ab)\omega_f. \quad (13)$$

The system is stable if the real parts of all three eigenvalues $\text{Re}\lambda_{1,2,3}$ are negative. The results of the numerical solution of the characteristic equation (12) are shown in Fig. 1, in dependence of the coupling coefficient, for chosen system parameters $a = 6$, $b = 0.1$, $\omega_f = 0.1$. The largest eigenvalues $\text{Re}\lambda$ cross zero and become negative at $k = 5.9$. The optimal values of the coupling coefficient k_{opt} , providing the highest rate ($\lambda = -0.1$) of convergence to the stabilized steady state, are from $k \approx 6.5$ to $k = 10$.

The necessary and sufficient conditions for stabilization can be found using the Hurwitz matrix

$$H = \begin{pmatrix} h_3 & h_1 & 0 \\ 1 & h_2 & 0 \\ 0 & h_3 & h_1 \end{pmatrix}. \quad (14)$$

According to the Routh-Hurwitz stability criterion the eigenvalues $\text{Re}\lambda_{1,2,3}$ are all negative if the diagonal minors of the H matrix are all positive

$$\Delta_1 = h_3 > 0, \quad \Delta_2 = h_3 h_2 - h_1 > 0, \quad \Delta_3 = h_1 \Delta_2 > 0. \quad (15)$$

Since Δ_2 should be positive according to the second inequality, the third inequality for Δ_3 can be replaced simply with $h_1 > 0$. This can be further simplified to $(1 - ab) > 0$ since $\omega_f > 0$ by definition. We note that, due to inequality (4), the last inequality will always be satisfied. Consequently, we are left with the

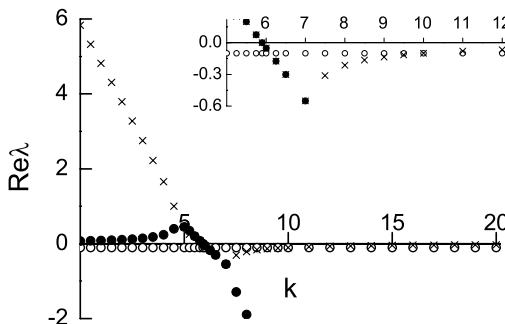


Fig. 1. Dependence of the $\text{Re}\lambda$ on the coupling coefficient k from Eq. (12). $a = 6$, $b = 0.1$, $\omega_f = 0.1$. The insert is a zoomed view of the $\text{Re}\lambda$ in the k range from 5 to 12.

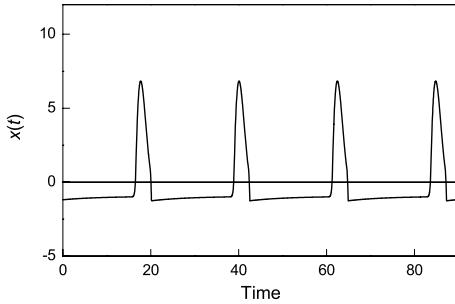


Fig. 2. Spikes $x(t)$ from Eq. (1-3). $a = 6$, $b = 0.1$, $\xi = 1.7$, $d_1 = 60$, $d_2 = 7$.

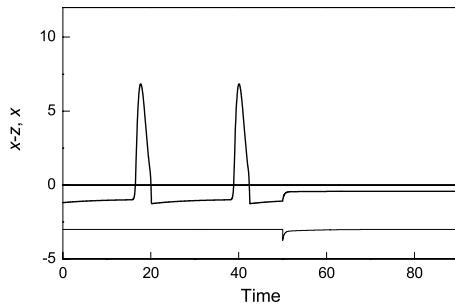


Fig. 3. Main variable $x(t)$ (top trace) and control signal $x - z$ (bottom trace) in the case the control is switched on between two spikes. The control signal display is lowered for the sake of clarity. $a = 6$, $b = 0.1$, $\xi = 1.7$, $d_1 = 60$, $d_2 = 7$, $\omega_f = 0.1$, $k = 9$.

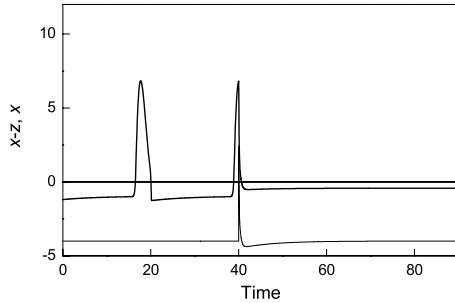


Fig. 4. Main variable $x(t)$ (top trace) and control signal $x - z$ (bottom trace) in the case the control is switched on during a spike. Parameters the same as in Fig. 3.

first and the second inequality of (15). We make an estimation of the threshold coupling coefficient k_{th} by requiring that for $k > k_{th}$, the two minors $\Delta_{1,2}$ should be positive. The result $k_{th} = 5.9$ obtained for the system parameters is in good agreement with the corresponding value found from $\text{Re}\lambda(k)$ in Fig. II. Results of

the numerical integration of Eqs. (1–3) as well as of Eqs. (8–10) are shown in Fig. 2, in Fig. 3, and in Fig. 4.

The waveform in Fig. 2 generated by the AFHN model more closely resembles the behavior of the spiking neurons in the HH model than that of the classical FHN model with symmetric activation function. When the control ($k > k_{th}$) is turned on, the spikes are totally suppressed and the system is stabilized on non-zero steady state x_0 , where the control signal $x - z$ vanishes. The transient process is extremely short: after control is on, only one short spike emerges.

3 Analogue Circuit and Experimental Results

The electrical circuit imitating the dynamics of the AFHN system along with the RC adaptive controller is sketched Fig. 5. The element values of the electronic neuron cell were chosen as $R_1 = R_2 = 1 \text{ k}\Omega$, $R_3 = R_6 = 200 \Omega$, $R_4 = 30 \Omega$, $R_5 = 240 \Omega$, $R_7 = 620 \Omega$, $C = 330 \text{ nF}$, $L = 1 \text{ H}$. The diodes D1 and D2 are the D1N5820 or similar type Schottky devices (the forward voltage drop $V^* \approx 0.2 \text{ V}$ at 1 mA). The operational amplifier OA is the LM741 type IC. The inductor L was implemented as an active gyrator subcircuit [19]. The element values of the controller were $R^* = 200 \Omega$, $C^* = 22 \mu\text{F}$. Using the Kirchhoff's laws the following equations for the circuit in Fig. 5 can be written:

$$C \frac{dV_C}{dt} = \frac{V_C}{R_3} - \frac{V_C}{R_7} - I_d(V_C) - I_L - \frac{V_0}{R_3} - \frac{V_C - V_{C^*}}{R^*}, \quad (16)$$

$$L \frac{dI_L}{dt} = V_C - R_6 I_L, \quad (17)$$

$$C^* \frac{dV_{C^*}}{dt} = \frac{V_C - V_{C^*}}{R^*}, \quad (18)$$

where

$$I_d(V_C) = \begin{cases} (V_C + V^*)/R_4, & V_C < -V^* \\ 0, & -V^* \leq V_C \leq V^* \\ (V_C - V^*)/R_5, & V_C > V^* \end{cases}. \quad (19)$$

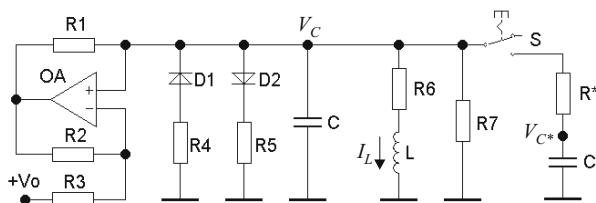


Fig. 5. Circuit diagram of the electronic neuron cell with an adaptive controller. $+V_0$ is a DC bias. The switch S, the resistor R^* and the capacitor C^* compose the controller.

By introducing the following set of dimensionless variables and parameters

$$x = \frac{V_C}{V^*}, \quad y = \frac{\rho I_L}{V^*}, \quad t = \frac{t}{\sqrt{LC}}, \quad \rho = \sqrt{\frac{L}{C}}, \quad a = \frac{\rho(R_7 - R_3)}{R_3 R_7},$$

$$b = \frac{R_6}{\rho}, \quad \xi = \frac{\rho V_0}{R_3 V^*}, \quad d_1 = \frac{\rho}{R_4}, \quad d_2 = \frac{\rho}{R_5}, \quad k = \frac{\rho}{R^*}, \quad \omega_f = \frac{\rho C}{R^* C^*}.$$

Here R^* is chosen to provide appropriate value of k , while C^* is chosen to ensure low enough value of the ω_f . Eqs. (16–18) can be presented in the form exactly coinciding with Eqs. (8–10), while Eq. (19) transforms into Eq. (3). Corresponding experimental results are presented in the Figs. 6–9. The spike train (Fig. 7) and the controlled dynamics (Figs. 8, 9) are in a good agreement with the numerical results, presented in the previous section.

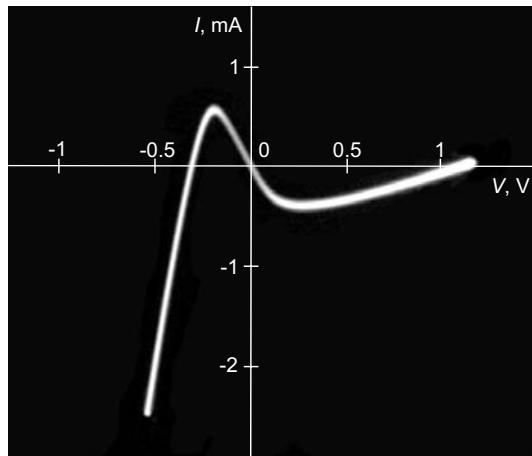


Fig. 6. DC Current-voltage characteristic of the electronic neuron cell. The elements C, L and R6 are removed from the circuit, the switch S is in the open position.

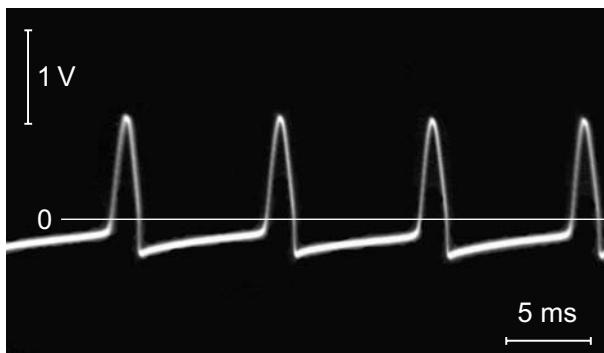


Fig. 7. Typical train of spikes $V_C(t)$ from the circuit without control

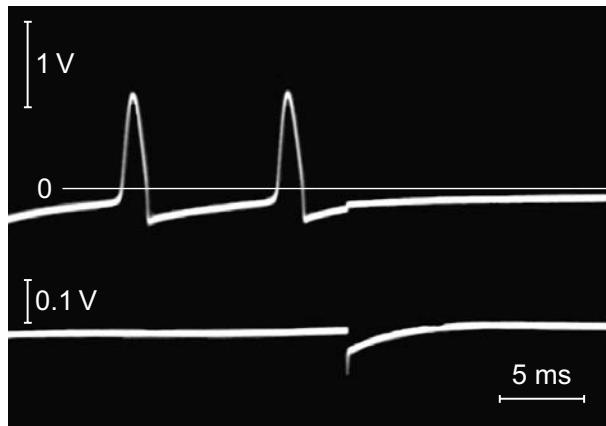


Fig. 8. Output voltage $V_C(t)$ (top trace) and control signal V_{R^*} (bottom trace) if the control is switched on during the interspike-interval. The control signal display is lowered for the sake of clarity.

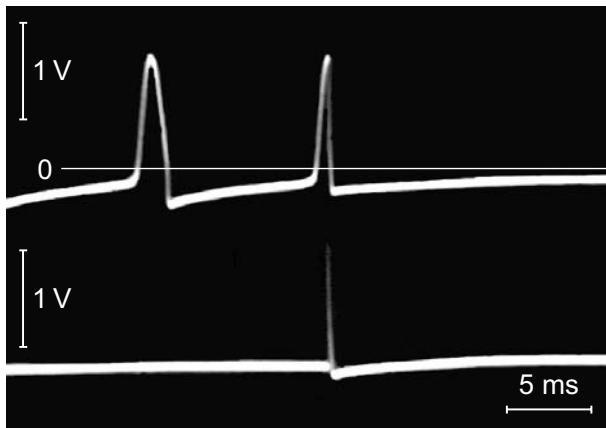


Fig. 9. Output voltage $V_C(t)$ (top trace) and control signal V_{R^*} (bottom trace) if the control is switched on during a spike. The control signal display is lowered for the sake of clarity.

4 Conclusions

In this work, a modification of the classical FitzHugh–Nagumo (FHN) model is proposed. The common symmetric activation function $x - x^3$ is replaced with a three-segment piecewise linear asymmetric function. In contrast to the common symmetric model at standard parameter values, the asymmetric version of the FHN model (AFHN) exhibits narrow triangular shaped spikes, similarly to more complicated biologically inspired Hodgkin–Huxley model. Besides, a nonlinear

electronic analog has been designed, built and investigated experimentally. Besides, we have applied the control method, based on the linear first order RC filter, to stabilize the unstable steady state of a spiking neuron. The method is fully adaptive; it automatically locates the steady state the coordinates of which can be left unevaluated.

Acknowledgments. The work was supported in part by the Lithuanian State Science and Studies Foundation.

References

1. Bielawski, S., Bouazaoui, M., Derozier, D., Glorieux, P.: Stabilization and Characterization of Unstable Steady State in a Laser. *Phys. Rev. A* 47, 3276–3279 (1993)
2. Johnson, G.A., Hunt, E.R.: Derivative Control of the Steady State in Chua's Circuit Driven in the Chaotic Region. *IEEE Trans. Circuit Syst. I* 40, 833–834 (1993)
3. Parmananda, P., Rhode, M.A., Johnson, G.A., Rollins, R.W., Dewald, H.D., Markworth, A.J.: Stabilization of Unstable Steady State in an Electrochemical System Using Derivative Control. *Phys. Rev. E* 49, 5007–5011 (1994)
4. Rulkov, N.F., Tsimring, L.S., Abarbanel, H.D.I.: Tracking Unstable Orbits in Chaos Using Dissipative Feedback Control. *Phys. Rev. E* 50, 314–324 (1994)
5. Namajūnas, A., Pyragas, K., Tamaševičius, A.: Stabilization of an Unstable Steady State in a Mackey-Glass System. *Phys. Lett. A* 204, 255–262 (1995)
6. Namajūnas, A., Pyragas, K., Tamaševičius, A.: Analog Techniques for Modeling and Controlling the Mackey-Glass System. *Int. J. Bif. Chaos* 7, 957–962 (1997)
7. Ciofini, M., Labate, A., Meucci, R., Galanti, M.: Stabilization of Unstable Fixed Points in the Dynamics of a Laser with Feedback. *Phys. Rev. E* 60, 398–402 (1999)
8. Schenck zu Schweinsberg, A., Dressler, U.: Characterization and Stabilization of the Unstable Fixed Points of a Frequency Doubled Nd: YAG Laser. *Phys. Rev. E* 63, 056210 (2001)
9. Huijberts, H.: Linear Controllers for the Stabilization of Unknown Steady States of Chaotic systems. *IEEE Trans. Circuits and Syst.-I* 53, 2246–2254 (2006)
10. Brown, D.J.: Adaptive Steady-State Stabilization for Nonlinear Dynamical Systems. *Phys. Rev. E* 78, 16213 (2008)
11. Rocşoreanu, C., Georgescu, A., Giugăeanu, N.: The FitzHugh-Nagumo Model: Bifurcation and Dynamics. Kluwer Academic Publishers, Dordrecht (2000)
12. Binczak, S., Kazantsev, V.B., Nekorkin, V.I., Bilbaut, J.M.: Experimental Study of Bifurcations in Modified FitzHugh-Nagumo Cell. *Electron. Lett.* 39, 961–962 (2003)
13. Aliaga, J., Busca, N., Mince, V., Mindlin, G.B., Pando, B., Salles, A., Sczupak, L.: Electronic Neuron within a Ganglion of a Leech (*Hirudo Medicinalis*). *Phys. Rev. E* 67, 061915 (2003)
14. Jacquir, S., Binczak, S., Bilbaut, J.M., Kazantsev, V.B., Nekorkin, V.I.: Study of Electronic Master-Slave MFHN Neurons. In: 12th International Workshop on Nonlinear Dynamics of Electronic Systems, pp. 182–185. Centro de Geofísica de Évora, Universidade de Évora, Évora (2004)
15. Jacquir, S., Binczak, S., Bilbaut, J.-M., Kazantsev, V., Nekorkin, V.: Synaptic Coupling Between Two Electronic Neurons. *Nonl. Dyn.* 44, 29–36 (2006)

16. Sitt, J.D., Aliaga, J.: Versatile Biologically Inspired Electronic Neuron. *Phys. Rev. E* 76, 051919 (2007)
17. Duan, S., Liao, X.: An Electronic Implementation for Liao's Chaotic Delayed Neuron Model with Non-Monotonous Activation Function. *Phys. Lett. A* 369, 37–43 (2007)
18. Malmivuo, J., Plonsey, R.: Bioelectromagnetism – Principles and Applications of Bioelectric and Biomagnetic Fields. Oxford University Press, New York (1995),
<http://www.bem.fi/book/>
19. Horowitz, P., Hill, W.: The Art of Electronics. Cambridge University Press, Cambridge (1993)

Combined Mechanisms of Internal Model Control and Impedance Control under Force Fields

Naoki Tomi, Manabu Gouko, and Koji Ito

Department of Computational Intelligence and Systems Science,
Tokyo Institute of Technology,

4259 Nagatsuta, Midori-ku, Yokohama, 226-8502 Japan

naoki@ito.dis.titech.ac.jp, {gouko,ito}@dis.titech.ac.jp

<http://www.ito.dis.titech.ac.jp>

Abstract. In daily life, humans must compensate for loads arising from interaction with the physical environment. Recent studies in human motor control have shown that when compensating for these loads, humans combine two feedforward control strategies, internal model control and impedance control. However, the combined mechanisms of the two control strategies have not been clarified. We propose a computational model of human arm movements and discuss how humans combine the two control strategies. We use an optimal regulator and simulate human arm movements under dynamic environments.

Keywords: Human motor control, internal model, impedance control, optimal control, arm movement.

1 Introduction

To manipulate objects or to use tools, humans must compensate for loads arising from interaction with the physical environment. Many psychophysical experiments for investigating motor adaptation of arm movements to external loads have been conducted [1]. These experiments have shown that humans use two types of feedforward control strategies when compensating loads [2]. One is “internal model control”, which humans use to compensate for the load by generating an opposing force to the external load [3]. To generate the opposing force, humans must learn the dynamics of the external load, which is called the “internal model” [4]. On the other hand, “impedance control”, is the alternate strategy in which humans regulate arm impedance (inertia, viscosity, and elasticity) depending on the task and compensate for the load by its impedance [5]. For skillful movements, humans must combine these two strategies depending on the task [6]. However, the mechanism in which humans combine two control strategies has not been clarified.

Many computational studies of arm movements have been performed [7]. In these studies, an optimal control scheme has often been used. An optimal controller calculates the motor command u , which minimizes the cost function J for

nonlinear arm dynamics $\dot{x} = f(x, u)$. When choosing certain parameters (ex. joint torque [7], hand force [8], variance of hand position [9] or reward [10]) as the cost function, optimized arm movements closely coincide with actual human movements. However, these computational models lack the modeling of the combined mechanism of two feedforward control strategies (which are the internal model control and impedance control) against external loads described above.

Therefore, the purpose of our study is to construct a computational model of arm movements, which enables discussion on the combined mechanism of internal model control and impedance control against external loads. For this purpose, we use an optimal regulator as the model and compare simulated arm movements with actual human movements. In section II, we formulate a computational model of arm movements using the optimal regulator. In sections III and IV, we discuss the point-to-point reaching experiments with the model and discuss the results.

2 Model

2.1 Outline of Model

Figure 1 shows a block diagram of a feedforward control scheme of human arm movements. In the musculoskeletal system, both the external force F_{ext} from the environment and the feedback force F_{imp} of the muscle impedance act upon the arm. The amplitude of the muscle impedance changes depending on the motor command u . ‘ J ’ and ‘ G ’ are Jacobian matrices, which coordinate transformation between the muscle, joint, and hand spaces.

We propose a computational model of a feedforward control scheme of human arm movements. In our model, the controller is composed of two units, the optimal controller and forward model, as shown in Fig. 1. The forward model works as a predictor of arm movements. The efferent copy of motor command u is inputted to the forward model, which calculates the predicted state \hat{q} of the arm. The optimal controller calculates the optimal motor command u , which drives the arm movement toward the desired state q_d . Both q_d and \hat{q} are inputted to the optimal controller. We can formulate the optimal control problem of this system as follows.

$$\dot{x} = f(x, u) \quad (1)$$

$$J = P(x) + \int_0^{t_f} L(x, u) dt \quad (2)$$

Equation (1) represents the state equation of the arm, and the state variable x represents the joint angle q and joint angle velocity \dot{q} . The control input u represents the motor command to the muscle. Equation (2) represents the cost function, which is the function of x and u . The constant t_f represents the termination time of the movement. The optimal controller calculates the optimal motor command u , which minimizes the cost function J under the boundary condition. In this formulation, there are several algorithms for optimization. We

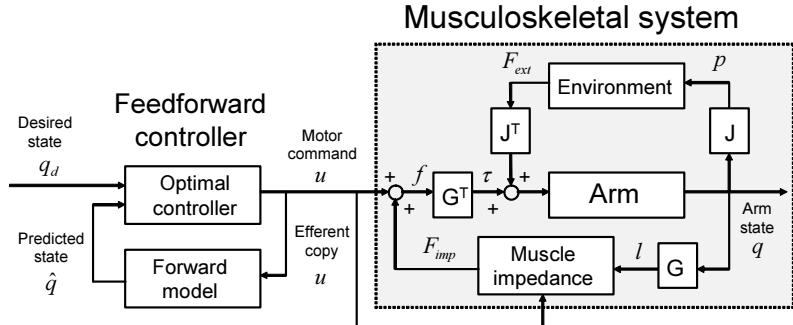


Fig. 1. Feedforward control scheme of human arm movements

model the controller as the optimal regulator, which is one of the simplest algorithms for optimization. The optimal regulator controls the system as \hat{q} coincides with q_d . In the following section, we derive the state equation $\dot{x} = f(x, u)$ of the musculoskeletal system. Then, we locally linearize the nonlinear state equation near the equilibrium point. Finally, we determine the motor command u of the optimal regulator using the linearized model.

2.2 Musculoskeletal System

Figure 2 shows the musculoskeletal system of the arm used in our study. We model the arm as two links and use two joints and six muscles in which six muscles are attached to the shoulder and elbow joints. The muscle tension generated by each muscle is modeled as follows.

$$f' = u + Bl + K(l - l_e), \quad (3)$$

$$T\dot{f} + f = f', \quad (4)$$

where $f = [f_1, f_2, \dots, f_6]^T$, $l = [l_1, l_2, \dots, l_6]^T$, and $u = [u_1, u_2, \dots, u_6]^T$ represent the muscle tension, muscle length and motor command of each muscle, respectively, l_e represents the equilibrium point of each muscle, $B = \text{diag}(b_1, b_2, \dots, b_6)$ and $K = \text{diag}(k_1, k_2, \dots, k_6)$ represent the viscoelastic parameters of each muscle, and T represents the time constant. The muscle tension f is smoothed by using a low-pass filter (shown in Eq. (4)). The joint torques $\tau = [\tau_1, \tau_2]^T$ of the shoulder and elbow are represented as a function of f as follows.

$$\tau = -G^T f, \quad (5)$$

where

$$G = \begin{bmatrix} -d_1 & d_1 & 0 & 0 & -d_3 & d_3 \\ 0 & 0 & -d_2 & d_2 & -d_4 & d_4 \end{bmatrix}^T \quad (6)$$

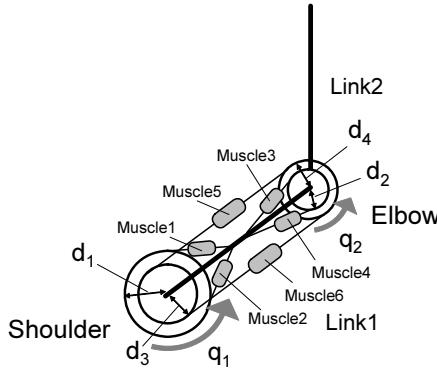


Fig. 2. Arm model (two joints - six muscles)

represents the Jacobian, and d_1 , d_2 , d_3 , and d_4 are the moments of the arm muscles.

The dynamic equation of the arm is represented as the following differential equation.

$$M(q)\ddot{q} + C(q, \dot{q}) = \tau + J^T F_{ext}, \quad (7)$$

where $q = [q_1, q_2]^T$ is the joint angle, M is the inertia of the arm, C is the coriolis/centripetal force, J is the Jacobian, and F_{ext} is the external load from the environment. The above equation transforms into the state equation with the muscles dynamics as follows.

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \\ \dot{f} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1}(q)(-C(q, \dot{q}) + \tau + J^T F_{ext}) \\ T^{-1}(-f + f') \end{bmatrix} \quad (8)$$

When we set the state variable as $x = [q_1, q_2, \dot{q}_1, \dot{q}_2, f_1, f_2, \dots, f_6]^T$, Eq. (8) becomes the nonlinear differential equation $\dot{x} = f(x, u)$.

2.3 Local Linearization of Nonlinear Dynamics

The muscle tension f' in Eq. (3) and the external force F_{ext} from the environment are represented as the function of q and \dot{q} as follows.

$$\begin{aligned} f' &= u + Bl + K(l - l_e) \\ &= u + BG\dot{q} + KG(q - q_e) \end{aligned} \quad (9)$$

$$\begin{aligned} F_{ext} &= B_{ext}\dot{p} + K_{ext}p \\ &= B_{ext}J\dot{q} + K_{ext}g(q) \end{aligned} \quad (10)$$

In Eq. (9), q_e is the equilibrium point of q , p is the hand position in the work space, and B_{ext} and K_{ext} represent the viscoelastic parameters of the environments. (In this transformation, we used the kinematic transformations $\dot{l} = G\dot{q}$, $l = Gq + l'$, $\dot{p} = J\dot{q}$, and $p = g(q)$. The constant l' represents the muscle length when $q = 0$.) By using Eqs. (5), (8), (9), (10), and the following equation,

$$\Delta\dot{x} = \frac{\partial f}{\partial x}(x_o, u_o)\Delta x + \frac{\partial f}{\partial u}(x_o, u_o)\Delta u, \quad (11)$$

we can locally linearize the nonlinear dynamics of the arm near the equilibrium point. The constants x_o and u_o represent the equilibrium point.

2.4 Formulation of Optimal Regulator

The motor command u using the optimal regulator is calculated by using the locally linearized model and solving the finite-time optimal control problem as follows.

$$\dot{x} = Ax + Bu \quad (12)$$

$$J = x^T Q_f x + \int_0^{t_f} (x^T Q x + u^T R u) dt \quad (13)$$

Equation (12) represents the state equation locally linearized ($A = \frac{\partial f}{\partial x}(x_o, u_o)$ and $B = \frac{\partial f}{\partial u}(x_o, u_o)$). Equation (13) represents the cost function, where Q_f , Q , and R are the weights and t_f is the termination time of the movement. The optimal feedback gain is represented as follows.

$$K_{gain} = -R^{-1}B^T P(t), \quad (14)$$

where $P(t)$ is the solution of the Riccati equation as follows.

$$-\dot{P} = A^T P + PA - PBR^{-1}B^T P + Q \quad (15)$$

$$P(t_f) = Q_f \quad (16)$$

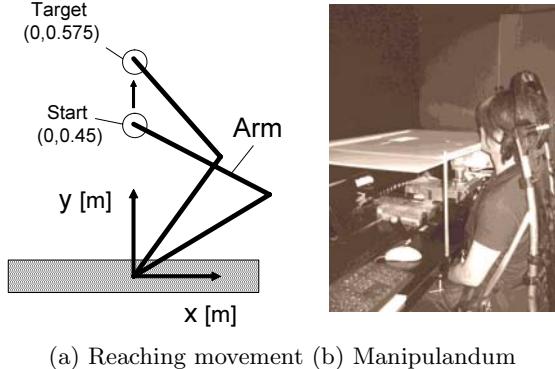
In our model, the controller calculates the optimal feedback gain K_{gain} as the predicted state \hat{x} from the forward model coincides with the desired state x_d . Thus, the motor command u is represented as follows.

$$u = -R^{-1}B^T P(t)(\hat{x} - x_d) \quad (17)$$

3 Simulation

3.1 Experimental Setup

In this section, we compare the arm movements using the optimal regulator with actual human arm movements. As a task, we tested a point-to-point arm

**Fig. 3.** Simulation task**Table 1.** Link parameters

	Link1	Link2
Mass [kg]	1.59	1.44
Link length [m]	0.3	0.35
Moment of inertia [kgm^2]	0.18	0.21
Center of mass [m]	6.78×10^{-2}	7.99×10^{-2}

movement in the horizontal plane [3], as shown in Fig. 3(a). The point-to-point arm movement is a task in which humans move their right hands from the starting point (0,0.45) toward the target point (0,0.575) is shown in Fig. 3(a). The movement time t_f was 300 ms and we measured these movements using robotic manipulandum [6], as shown in Fig. 3(b).

The simulation parameters of mass, link length, moment of inertia, and center of mass were set up, as shown in Table 1. The moments of the arm muscles were $(d_1, d_2, d_3) = (0.04, 0.025, 0.035)$ m. The viscoelastic parameters of the muscles were $B = \text{diag}(2.5, 2.5, 2.5, 2.5, 2.5, 2.5) \times 10^3$ Ns/m and $K = \text{diag}(2, 2, 1, 1, 1, 1) \times 10^3$ N/m. We locally linearized the arm dynamics near the starting point of the movement (0,0.45) and calculated the output of the optimal regulator using the linear model obtained from the linearization. The parameters of the cost function were $Q_f = \text{diag}(2 \times 10^6, 5 \times 10^5, 5 \times 10^4, 5 \times 10^4, 10, 10, 10, 10, 10, 10)$, and $Q = 0$ and $R = \text{diag}(1, 1, 1, 1, 1)$.

The purpose of this experiment was to test the simulated movements by the optimal regulator and compare them with the actual human movements. For simplicity, we assumed that the forward model can be used to perfectly predict arm dynamics.

3.2 Point-to-Point Movements under Null Field

Figure 4 shows the simulation results of point-to-point arm movements under Null field (Eq. (10): $B_{ext} = 0$, $K_{ext} = 0$). Each figure (Figs. 4(a)-(d)) represents

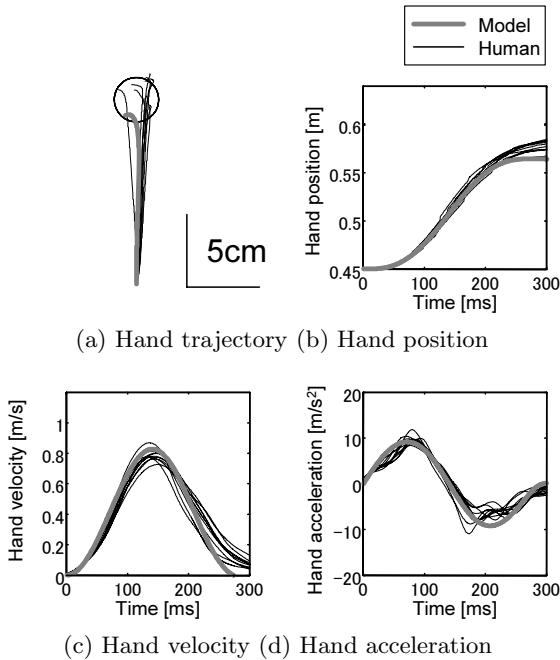


Fig. 4. Reaching movements under null field

(a) hand trajectories in the x-y plane, (b) time-series data of hand position, (c) time-series data of hand velocity and (d) time-series data of hand acceleration. The thick lines represent the simulation results of the model and the thin lines represent human movement data ($N=1$). It is clear that the simulation results coincide with actual human movements. In the model prediction, the hand reaches the target position in 300 ms and the bell-shape hand velocity and sine-wave hand acceleration patterns are generated.

3.3 Point-to-Point Movements under Force Field

Figure 5 shows the simulation results of point-to-point arm movements under force fields. The force fields apply loads to the hand during movement, and the loads are proportional to the hand position and hand velocity. We set up a velocity-dependent force field (Eq. (10): $B_{ext} = [0 \ -25; 25 \ 0]$, $K_{ext} = 0$) [6]. In this force field, the load was applied in the left direction toward the target position, as shown in Fig. 5. Figure 5(a) shows human hand trajectories ($N=5$). At first, the subjects were not able to perform straight movements because of the load (Before learning). However, as the subjects repeated the movements, they learned to compensate for the load and finally they were able to perform straight movements (After learning). In addition, when the subjects performed movements in Null field after learning, hand trajectories curved in the opposite

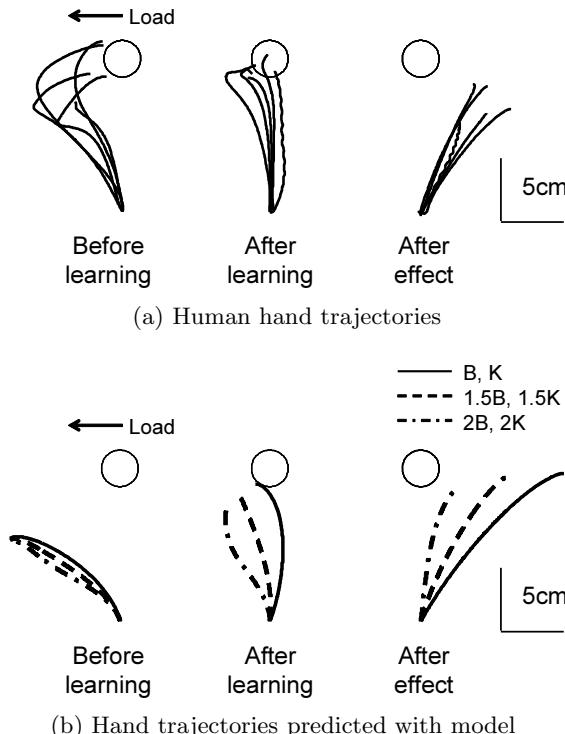


Fig. 5. Reaching movements under velocity-dependent force field

direction of the load (after effect). Figure 5(b) shows the simulation results of the model. The solid lines almost coincide with actual human movements. (In this simulation, we used the motor commands calculated in Null field before learning and the motor commands calculated in the force field in the after-effect trial.) The dashed lines represent the simulation results when the muscle impedance parameters B and K were set up doubled. In a high impedance condition, the hand trajectories curved after learning and the curvature in the after-effect trial decreased. This indicates that when muscle impedance increases, humans combine impedance control with internal model control to compensate for the load.

4 Discussion

Recently, optimal regulators have been used for computational modeling of arm movements for investigating motor control mechanisms under noisy and uncertain environments [11, 12, 13, 14]. In these studies, the arm dynamics are represented as a stochastic differential equation. On the other hand, in our study, we used the local linearized model of the arm and simulated arm movements

under dynamic environments. As a result, we confirmed that (1) it is possible to generate humanlike arm movements in the Null and force fields by using a local linearized optimal regulator for a nonlinear system and (2) the optimized movements are changed by the muscle impedance parameters.

As shown in Fig. 5(b), when the muscle impedance increases, the optimal controller generates curved hand trajectories after learning and the curvature in the after-effect trial decreases. This indicates that humans combine impedance control with internal model control to compensate the load. Thus, it is assumed that internal model control and impedance control work together depending on the task and are optimally combined. Some psychophysical experiments of arm movements support this result [5, 6, 15, 16, 17]. These experiments show that humans learn optimal arm impedance under dynamic environments.

In future studies, we need to construct more sophisticated muscle models. In this paper, the muscle impedance was set as a constant. However, the amplitude of the muscle impedance changes depending on the motor command. Thus, in future work, we plan to model the arm as a “bilinear model” to follow this characteristic and calculate the output of the optimal regulator using a bilinear model [18]. In addition, we need to add a feedback control system to our model. To add this feedback control system, it is important to take into account the time delay of sensory feedback signals [19].

5 Conclusion

We proposed a computational model of human arm movements and discussed how humans combine internal model control and impedance control. We used an optimal regulator and simulated human arm movements under dynamic environments. As a result, we confirmed that the optimal regulator is able to generate humanlike arm movements and the optimized movements are changed by the muscle impedance parameters. In future work, we plan to simulate arm movements using a bilinear optimal control.

Acknowledgments. Part of this research was supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research on Priority Areas (No.454) and Scientific Research (B) (No.21360201).

References

1. Shadmehr, R., Wise, S.P.: *The Computational Neurobiology of Reaching and Pointing*. The MIT Press, Cambridge (2005)
2. Franklin, D.W., Osu, R., Burdet, E., Kawato, M., Milner, T.E.: Adaptation to stable and unstable dynamics achieved by combined impedance control and inverse dynamics model. *Journal of Neurophysiology* 90, 3270–3282 (2003)
3. Shadmehr, R., Mussa-Ivaldi, F.A.: Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience* 14(5), 3208–3224 (1994)

4. Jordan, M.I., Rumelhart, D.E.: Forward models: Supervised learning with a distal teacher. *Cognitive Science* 16, 307–354 (1992)
5. Burdet, E., Osu, R., Franklin, D.W., Milner, T.E., Kawato, M.: The central nervous system stabilizes unstable dynamics by learning optimal impedance. *Nature* 414, 446–449 (2001)
6. Tomi, N., Gouko, M., Ito, K.: Inaccuracy of internal models in force fields and complementary use of impedance control. In: *Proceedings of 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, pp. 393–398 (2008)
7. Uno, Y., Kawato, M., Suzuki, R.: Formation and control of optimal trajectory in human multi-joint arm movement-minimum torque change model. *Biological Cybernetics* 61, 89–101 (1989)
8. Ohta, K., Svinin, M.M., Luo, Z.W., Hosoe, S., Laboissiere, R.: Optimal trajectory formation of constrained human arm reaching movements. *Biological Cybernetics* 91, 23–36 (2004)
9. Harris, C.M., Wolpert, D.M.: Signal-dependent noise determines motor planning. *Nature* 394, 780–784 (1998)
10. Izawa, J., Kondo, T., Ito, K.: Biological arm motion through reinforcement learning. *Biological Cybernetics* 91, 10–22 (2004)
11. Todorov, E., Jordan, M.I.: Optimal feedback control as a theory of motor coordination. *Nature Neuroscience* 5(11), 1226–1235 (2002)
12. Li, W., Todorov, E.: Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic systems. *International Journal of Control* 80(9), 1439–1453 (2007)
13. Izawa, J., Rane, T., Donchin, O., Shadmehr, R.: Motor adaptation as a process of reoptimization. *Journal of Neuroscience* 28(11), 2883–2891 (2008)
14. Izawa, J., Shadmehr, R.: On-line processing of uncertain information in visuomotor control. *Journal of Neuroscience* 28(44), 11360–11368 (2008)
15. Osu, R., Burdet, E., Franklin, D.W., Milner, T.E., Kawato, M.: Different mechanisms involved in adaptation in stable and unstable dynamics. *Journal of Neurophysiology* 90, 3255–3269 (2003)
16. Takahashi, C.D., Scheidt, R.A., Reinkensmeyer, D.J.: Impedance control and internal model formation when reaching in a randomly varying dynamical environment. *Journal of Neurophysiology* 86, 1047–1051 (2001)
17. Liu, J., Reinkensmeyer, D.: Motor adaptation as an optimal combination of computational strategies. In: *Proceedings of the 26 th Annual International Conference of the IEEE EMBS*, pp. 4025–4028 (2004)
18. Hofer, E.P., Tibken, B.: An iterative method for the finite-time bilinear-quadratic control problem. *Journal of Optimization Theory and Applications* 57(3), 411–427 (1988)
19. Bhushan, N., Shadmehr, R.: Computational nature of human adaptive control during learning of reaching movements in force fields. *Biological Cybernetics* 81, 39–60 (1999)

Neural Network Control of Unknown Nonlinear Systems with Efficient Transient Performance

Elias B. Kosmatopoulos, Diamantis Manolis, and M. Papageorgiou

Dynamic Systems and Simulation Laboratory
Department of Production & Management Engineering
Technical University of Crete
Chania 73100, Greece
Tel.: +30-28210-37306
`{kosmatop,diamantis,markos}@dssl.tuc.gr`

Abstract. In this paper we provide a neural-based semi-global stabilization design for unknown nonlinear state-feedback stabilizable systems. The proposed design is shown to guarantee arbitrary good transient performance outside the regions where the system is uncontrollable. This is made possible through an appropriate combination of recent results developed by the author in the areas of adaptive control and adaptive optimization and a new result on the convex construction of Control Lyapunov Functions (CLF) for nonlinear systems.

Keywords: High Order Neural Networks (HONN), Persistence of Excitation (PE), Exploitation vs Exploration, Control Lyapunov Function (CLF), Sum of Squares (SoS).

1 Introduction

Recently, we introduced a new adaptive/learning methodology applicable to adaptive optimization of unknown functions and adaptive fine-tuning of large-scale nonlinear control systems [6][7]. The key idea behind this methodology is the use of *concurrent exploitation-exploration*: each time a control action should be taken, a set of randomly-generated *candidate-control actions* is generated and the best one – as estimated by appropriate neural estimators – is chosen. Using the above simple logic, the control actions are guaranteed to provide both with exploitation (in the sense that they are “moving” in the right directions) and exploration (i.e. they provide with PE). Moreover, as it was seen in [6][7], if the random generation of candidate control actions is appropriately done, then the concurrent exploitation-exploration methodology guarantees efficient performance and convergence.

The concurrent exploitation-exploration methodology of [6][7] has also been successfully extended to the case of stabilization of a class of unknown nonlinear systems [8]. However, contrary to the case of adaptive optimization and adaptive fine-tuning where the problem is to optimize an available for measurement, but otherwise unknown, objective function this is not the case in stabilization of

unknown nonlinear systems. In the latest case, the “objective” function to be optimized corresponds to the so-called Control Lyapunov Function (CLF), see [14,15], which, in general, is not available for measurement in the case of unknown nonlinear systems. For this reason, the approach of [8] is limited to unknown systems for which a known CLF function (that may depend on the unknown system parameters) is available.

In this paper, we show that the concurrent exploitation-exploration methodologies of [6,7,8] appropriately combined with a new result on the construction of CLFs for known nonlinear systems can lead to an efficient, globally convergent, control methodology for general unknown nonlinear systems. According to this aforementioned new result, which is presented in section IV of this paper, the problem of constructing an approximate CLF for a general nonlinear system can be casted as a convex optimization problem. By combining this result with the concurrent exploitation-exploration methodologies of [6,7,8] we develop an adaptive/learning controller consisting of

- an adaptive HONN identifier for estimating the unknown system dynamics (see section III);
- a convex optimizer that on-line constructs the system CLF based on the estimation of the system dynamics generated by the HONN identifier;
- a concurrent exploitation-exploration control logic that chooses among many randomly generated control actions the one that leads to the maximum decrease of the system CLF as constructed in the previous step.

Using rigorous arguments we establish that the proposed methodology can guarantee closed-loop stability and arbitrarily good performance outside the regions where the system becomes uncontrollable, for any nonlinear system that is state-feedback stabilizable and satisfies an additional mild controllability assumption. Finally, simulation experiments performed on a challenging nonlinear control problem verify our theoretical results.

1.1 Notations and Preliminaries

I_n denotes the n -dimensional identity matrix. For a vector $x \in \mathbb{R}^n$, $|x|$ denotes the Euclidean norm of x (i.e., $|x| = \sqrt{x^\tau x}$), while for a matrix $A \in \mathbb{R}^{n^2}$, $|A|$ denotes the Frobenius norm of A . A function f is said to be C^m , where m is a positive integer, if it is uniformly continuous and its first m derivatives are uniformly continuous. The notation $\text{vec}(A, B, C, \dots)$, where A, B, C, \dots are scalars, vectors or matrices, is used to denote a vector whose elements are the entries of A, B, C, \dots (taken column-wise). For a symmetric matrix A , the notation $A \succ 0$ ($A \succeq 0$) is used to denote that A is a positive definite (resp. positive semidefinite) matrix. We say that a function $\chi : \mathbb{R}_+ \mapsto \mathbb{R}_+$ is of class \mathcal{K} (symbolically, $\chi \in \mathcal{K}$) when χ is continuous, strictly increasing, and $\chi(0) = 0$; we say that a function $\chi : \mathbb{R}_+ \mapsto \mathbb{R}_+$ is of class \mathcal{K}_∞ (symbolically, $\chi \in \mathcal{K}_\infty$) if it is of class \mathcal{K} and, moreover, $\lim_{r \rightarrow \infty} \chi(r) = \infty$. In order to avoid the usage of too many variables and constants, we will also make use of the following notation:

consider a subset $\mathcal{X} \subseteq \mathbb{R}^n$ and let $f_c : \mathbb{R}^n \mapsto \mathbb{R}^m$ be a function parameterized by a positive constant c ; the notation $f_c(x) = \mathcal{O}_{\mathcal{X}}(c)$ will be used throughout this paper – sometimes with a slight abuse of notation – if there exists a function $\chi \in \mathcal{K}$ such that $|f_c(x)| \leq \chi(c), \forall x \in \mathcal{X}$. We close this section by providing some definitions and a lemma that will be proved useful in the analysis of the main results of this paper.

Definition 1. Consider a subset $\mathcal{A} \subset \mathbb{R}^n$ of non-zero Lebesgue measure (which is not necessarily connected) and suppose that the random vector $x \in \mathbb{R}^n$ is generated according to a probability density function f . We say that f is a strictly-positive probability distribution over the subset \mathcal{A} if $\int_{\mathcal{A}} f(x)dx = 1$, $f(x) > 0$ for all $x \in \mathcal{A}$.

Definition 2. The matrix function $f : \mathbb{R}^n \mapsto \mathbb{R}^{L \times m}$ is said to be a monomial of size $L \times m$ (symbolically, $f \in \mathcal{M}_n^{L \times m}$) if the entries of f take the form $f_{ij}(x) = x_1^{a_{1,ij}} x_2^{a_{2,ij}} \cdots x_n^{a_{n,ij}}, i \in \{1, \dots, L\}, j \in \{1, \dots, m\}$ for some nonnegative integers $a_{1,ij}, a_{2,ij}, \dots, a_{n,ij}$ such that any $i, \iota \in \{1, \dots, L\}, i \neq \iota$, we have that $[a_{1,ij}, a_{2,ij}, \dots, a_{n,ij}] \neq [a_{1,\iota j}, a_{2,\iota j}, \dots, a_{n,\iota j}]$.

Definition 3. A matrix function $p : \mathbb{R}^n \mapsto \mathbb{R}^{k \times m}$ is said to be a polynomial function if there exists a positive integer L such that $p(x) = \theta f(x)$ for some $f \in \mathcal{M}_n^{L \times m}$ and a constant matrix $\theta \in \mathbb{R}^{k \times L}$. Let also \mathcal{X} be a subset of \mathbb{R}^n ; a polynomial matrix function $p \in \mathcal{P}_n^1$ is said to be an SoS polynomial over \mathcal{X} if there exist $p_1, p_2, \dots, p_\kappa \in \mathcal{P}_n^1$ such that $p(x) = \sum_{i=1}^{\kappa} p_i^2(x), \forall x \in \mathcal{X}$. Finally, a polynomial matrix function $p \in \mathcal{P}_n^1$ is said to be a Positive Definite (PD) polynomial over \mathcal{X} (symbolically $p \in PD_n(\mathcal{X})$) if $p(x) \geq 0, \forall x \in \mathcal{X}$ and $p(x) = 0 \iff x = 0$.

Lemma 1. [110] SoS polynomials are dense in the space of positive definite polynomials.

2 Problem Formulation

Let us consider the problem of constructing *semi-global state-feedback stabilizers* for a general *unknown* nonlinear system of the form

$$\dot{x} = F(x) + g(x)u, \quad x(0) \in \mathcal{X} \quad (1)$$

where $x \in \mathbb{R}^n, u \in \mathbb{R}^m$ denote the vectors of systems states and control inputs, respectively, F, g are continuous unknown nonlinear vector-fields of appropriate dimensions, with $F(x)$ satisfying $F(0) = 0$ and \mathcal{X} is a compact subset of \mathbb{R}^n containing the origin. The subset \mathcal{X} refers to the subset of all admissible initial conditions. It is worth noticing that, contrary to the problem of global stabilization in which case $\mathcal{X} \equiv \mathbb{R}^n$, in the case of semi-global stabilization the subset of initial states is assumed to be a bounded set which is *a priori* known.

In order to have a well-posed problem we will assume that the system (1) is state feedback stabilizable, i.e. we will assume that

(A1): There exists a continuous function $k : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that the solutions $x(t)$ of the closed-loop system $\dot{x} = F(x) + g(x)k(x)$ converge to zero asymptotically, for all $x_0 \in \mathbb{R}^n$.

According to standard results from the theory of nonlinear control, see e.g. [14, 15], assumption (A1) implies the existence of a C^1 , positive definite, radially unbounded function CLF $V : \mathbb{R}^n \mapsto \mathbb{R}_+$ which satisfies

$$\inf_u \left\{ \frac{\partial V^\tau}{\partial x} (F(x) + g(x)u) \right\} < 0, \forall x \neq 0 \quad (2)$$

Associated to the subset \mathcal{X} of admissible initial conditions we will define a “sufficiently large” compact subset $\bar{\mathcal{X}}$ which contains \mathcal{X} . Without loss of generality we will assume that the subset $\bar{\mathcal{X}}$ takes the following form $\bar{\mathcal{X}} = \{x : \mathbb{R}^n : |x| \leq \Omega\}$, where Ω is a sufficiently large positive constant so that \mathcal{X} lies in the interior of $\bar{\mathcal{X}}$ and, moreover, the distance between the boundaries of \mathcal{X} and $\bar{\mathcal{X}}$ is sufficiently large.

In the development and analysis of the proposed scheme we will need the following lemma that relates the CLF to the uncontrollable region of system (I):

Lemma 2. *Assumption (A1) implies that there exist positive constants ϵ_i , $i = 1, 2, 3$ such that, for all $x \in \bar{\mathcal{X}}$, $\left| \frac{\partial V^\tau}{\partial x}(x)g(x) \right| < \epsilon_1$ and $|x| > \epsilon_3$ imply that $\frac{\partial V^\tau}{\partial x}(x)F(x) < -\epsilon_2$.*

Proof. The proof is a directly corollary of Lemma 1 of [5].

Let us define the *uncontrollable region* of (I) to be the subset \mathcal{U} defined according to

$$\mathcal{U} = \left\{ x \in \mathbb{R}^n : |x| > \epsilon_3 \text{ and } \left| \frac{\partial V^\tau}{\partial x}(x)g(x) \right| < \epsilon_1 \right\}$$

Note that Lemma 2 implies that as long as $x(t) \in \mathcal{U}$, the choice $u(t) = 0$ guarantees that $V(t)$ is decreasing.

We close this subsection by introducing a second assumption on system (I):

(A2): Pick any $i \in \{1, \dots, n\}$ and let \mathcal{L} be any subset of $\mathbb{R}_+ \times \mathbb{R}$ which has zero Lebesgue measure. If $u(t)$ is a random vector generated according to a strictly-positive probability distribution over a subset of \mathbb{R}^m that has non-zero Lebesgue measure, then

$$\Pr \{x_i(t) \in \mathcal{L}\} = 0, \forall t \in \mathbb{R}_+$$

Assumption (A2) although it may appears too technical it can be seen that it is satisfied in a large number of control applications; see [8] for more details.

3 Adaptive HONNs for Learning the Unknown System Dynamics

As already mentioned in the Introduction, the proposed control scheme employs an adaptive neural structure for learning the unknown system (II) dynamics. It has to be emphasized that the problem of learning the unknown system (II) dynamics using neural networks is well-established, see e.g. [4][13]. In this paper we adopt the adaptive HONN structure of [4] appropriately modified to meet the requirements of the particular application. The details of the adaptive HONN structure used within the proposed control scheme are presented next.

Let L denote a user-defined positive integer which is assumed to be sufficiently large. Let us also fix two monomials $z_1, z_2 \in \mathcal{M}_n^L$ with z_1 satisfying¹

$$z_1(x) = 0 \text{ iff } x = 0 \quad (3)$$

Standard approximation results, see e.g. [4][11], can be used to establish the following lemma.

Lemma 3. *Fix the compact subset $\bar{\mathcal{X}} \subset \Re^n$. Then, there exist constant matrices $\theta_0, \theta_1, \dots, \theta_m \in \Re^{n \times L}$ so that the following hold:*

$$F(x) = \theta_0 z_1(x) + \mathcal{O}_{\bar{\mathcal{X}}} \left(\frac{1}{L} \right), \quad g_i(x) = \theta_i z_2(x) + \mathcal{O}_{\bar{\mathcal{X}}} \left(\frac{1}{L} \right) \quad (4)$$

where $g_i(x)$ denotes the i th column of $g(x)$.

Lemma 3 states that, given the monomials z_1, z_2 , there exists a set of HONN weight vectors $\theta_0, \theta_1, \dots, \theta_m$ such that the unknown system vector fields F, g can be approximated with arbitrary accuracy over the compact subset $\bar{\mathcal{X}}$. Based on lemma 3 and working similar to [4], we employ a learning scheme that combines a filtered regressor HONN and an integral adaptive learning law with projection for estimating the HONN weight vectors $\theta_0, \theta_1, \dots, \theta_m$. This learning scheme is described as follows: Let $\bar{\theta} = [\mathbf{a}, \theta_0, \theta_1, \dots, \theta_m]$ with $\mathbf{a} = [a, a, \dots, a]^T \in \Re^n$ and a being a user-defined positive constant and $\hat{\theta}_i(t)$ denote the estimate of the i th row of $\bar{\theta}$ as produced by the HONN estimator at time t . Then, the estimates $\hat{\theta}_i(t)$ are produced according to the following set of equations:

$$\dot{\varphi}_i = -a\varphi_i + [x_i, z_1^\tau(x), z_2^\tau(x)u_1, \dots, z_2^\tau(x)u_m]^\tau, \quad \dot{\hat{\theta}}_i = -\mathcal{P}_i \left(\gamma R_i \hat{\theta}_i + Q_i \right) \quad (5)$$

$$\dot{R}_i = -\beta R_i + \varphi_i \varphi_i^\tau, \quad R_i(0) = 0, \quad \dot{Q}_i = -\beta Q_i - (x_i - \hat{x}_i)\varphi_i, \quad Q_i(0) = 0 \quad (6)$$

$$\dot{\hat{x}}_i = \hat{\theta}_i^\tau \varphi_i$$

where \mathcal{P}_i denotes the standard projection operator, see e.g. [2], used to keep $\hat{\theta}_i(t)$ bounded and within a convex set and β, γ are positive user-defined constants. The estimate $\hat{\theta}(t) \in \Re^{n \times (m+1)L}$ of $\theta = [\theta_0, \theta_1, \dots, \theta_m]$ is then generated according to $\hat{\theta}(t) = [\hat{\theta}_0(t), \hat{\theta}_1(t), \dots, \hat{\theta}_m(t)]$, where $\hat{\theta}_0(t), \hat{\theta}_1(t), \dots, \hat{\theta}_m(t)$ are extracted from the estimate $\hat{\theta}(t)$ according to $\hat{\theta}(t) = [\hat{\mathbf{a}}(t), \hat{\theta}_0(t), \hat{\theta}_1(t), \dots, \hat{\theta}_m(t)]$.

¹ For simplicity, we assume that both monomials z_1, z_2 have the same size L ; all the results of this paper hold in case where the monomials z_1, z_2 have different size.

4 CLF Convex Design in the Case of Known System Dynamics

As already mentioned in the Introduction, the proposed control methodology makes use of a new result that shows that an approximate CLF for a general nonlinear system can be casted as a convex optimization problem. This result is formally presented in the next Theorem.

Theorem 1. *Let assumption (A1) hold and L be sufficiently large. Let also $M(x)$ denote the $L \times n$ polynomial matrix whose (i, j) th entry is given by $M_{ij}(x) = \frac{\partial z_{1i}}{\partial x_j}(x)$ where $z_{1i}(x)$ denotes the i th element of $z_1(x)$ and consider a collection of points $x_i, i = 1, \dots, N$ which are chosen randomly according to a uniform distribution in \mathcal{X} and let $\hat{P}, \hat{Q}, \hat{K}$ denote the matrix triplet that minimizes the following convex optimization problem*

$$\min_{\hat{P}, \hat{Q}, \hat{K}} \sum_{i=1}^N \left| J \left(\hat{P}, \hat{Q}, \hat{K}, x_i \right) \right|^2 \text{ subject to } \hat{P} \succ \epsilon_1 I_L, \hat{Q} \succ \epsilon_2 I_L \quad (7)$$

where ϵ_1, ϵ_2 are two positive user-defined constants and $J(\cdot)$ is defined according to

$$\begin{aligned} J(\bar{P}, \bar{Q}, K, x) &= z_1^\tau(x) \left(\bar{P} \theta_0^\tau M^\tau(x) + (\bar{g}(\theta, x) \bar{K})^\tau M^\tau(x) \right) z_1(x) \\ &\quad + z_1^\tau(x) \left(M(x) \theta_0 \bar{P} + M(x) \bar{g}(\theta, x) \bar{K} + \bar{Q} \right) z_1(x) \end{aligned}$$

where $\bar{g}(\theta, x) = [\theta_1 z_2(x), \dots, \theta_m z_2(x)]$. Then, there exist two positive constants $\epsilon_1^*, \epsilon_2^*$ such that, if $N \geq L(L+m+1)$, for any $\epsilon_1 \in (0, \epsilon_1^*], \epsilon_2 \in (0, \epsilon_2^*]$ the system (7) CLF satisfies

$$V(x) = z_1^\tau(x) \hat{P}^{-1} z_1(x) + \mathcal{O}_{\bar{\mathcal{X}}} \left(\frac{1}{L} \right), \text{ with probability 1} \quad (8)$$

Proof. Due to space limitations the proof is not included.

In the sequel we will denote with

$$\hat{P} = \mathbf{V}(\theta) \quad (9)$$

the solution \hat{P} of (7) as a function of θ .

5 The Proposed Scheme

Based on the information received by the learning mechanism (5), the proposed control scheme updates the control input u every Δt time-units; in other words, if $t_k = t_{k-1} + \Delta t, t_0 = 0, k \in \mathbb{Z}$ denote the time-instances at which the controller

vector is updated, then we have that $u(t)$ remains constant in the intervals $t \in [t_{k-1}^+, t_k]$. The proposed controller update scheme is as follows:

$$\begin{aligned} u(t_k^+) &= \arg \min_{\pm u_k^{(j)}, j \in \{1, \dots, m\}} \dot{\hat{V}}_k^{(\pm j)} \\ \mathcal{V}(\hat{\theta}_k, x_k) &= z_1^\tau(x_k) \mathbf{V}^{-1}(\hat{\theta}_k) z_1(x_k) \\ \dot{\hat{V}}_k^{(\pm j)} &= \frac{\partial \mathcal{V}^\tau}{\partial x}(\hat{\theta}_k, x_k) \hat{\theta}_k \phi(x_k, \pm u_k^{(j)}) \end{aligned} \quad (10)$$

where $\phi(x, u) = [z_1(x), z_2(x)u_1, \dots, z_2(x)u_m]$, $x_k = x(t_k)$, $\hat{\theta}_k = \hat{\theta}(t_k)$ and $u_k^{(j)}$ are m zero-mean random vectors in $[-2\alpha_k, -\alpha_k]^m \cup [\alpha_k, 2\alpha_k]^m$ satisfying

$$\left| \left[u_k^{(1)}, \dots, u_k^{(m)} \right] \right|^{-1} < \frac{\Xi}{\alpha_k} \quad (11)$$

where α_k is user-defined positive sequence and Ξ is a finite positive number independent of α_k . We are ready to establish the main result of this paper.

Theorem 2. *Let (A1), (A2) hold. Let also Δt be sufficiently small and L be sufficiently large. Then, for arbitrary $\bar{\alpha} > 0$, there exist finite positive constants $\beta_1, \beta_2, \gamma_1$ and a finite positive integer $\hbar = \mathcal{O}_{\bar{\mathcal{X}}} \left(\frac{1}{\gamma} \right)$ such that, if α_k satisfies*

$$\begin{aligned} \alpha_k \leq \beta_2 &\text{ if } \left| \frac{\partial \mathcal{V}^\tau}{\partial x}(\hat{\theta}_k, x_k) \bar{g}(\hat{\theta}_k, x_k) \right| < \hat{\epsilon}_1 \text{ or } k \leq \hbar \\ \alpha_k \geq \beta_1 &\text{ otherwise} \end{aligned} \quad (12)$$

where $\hat{\epsilon}_1$ is a positive design constant satisfying $\frac{1}{4}\epsilon_1 < \hat{\epsilon}_1 \leq \frac{1}{2}\epsilon_1$ and the adaptive gain γ of the estimator (5) satisfies , then, the proposed adaptive control scheme (5), (10), (11) guarantees that the closed-loop solutions are bounded and, moreover,

$$\limsup_{t \rightarrow \infty} |x(t)| \leq \epsilon_3, \text{ w.p.1} \quad (13)$$

and

$$\dot{V}(t_k^+) < -\bar{\alpha}, \text{ if } x_k \notin \mathcal{U} \text{ or } (x_k, \hat{\theta}_k) \notin \mathcal{S}_k, \text{ w.p.1} \quad (14)$$

where \mathcal{S}_k is a subset of $\Re^n \times \Re^{n \times 2L}$ that satisfies $\mathcal{S}_k = \emptyset, \forall k > \hbar$.

Proof. Due to space limitations the proof is not included.

Theorem 2 states that the proposed scheme guarantees arbitrarily good transient performance outside (a) the regions of the state space where the system becomes uncontrollable (i.e. for $x_k \notin \mathcal{U}$) and (b) the subset \mathcal{S}_k which shrinks exponentially fast (with rate of convergence proportional to $\mathcal{O}_{\bar{\mathcal{X}}} \left(\frac{1}{\gamma} \right)$): according to Theorem 2 the larger are the design terms α_k for $t_k : \left| \frac{\partial \mathcal{V}^\tau}{\partial x}(\hat{\theta}_k, x_k) \bar{g}(\hat{\theta}_k, x_k) \right| \geq \hat{\epsilon}_1$ and $k > \hbar$, the larger is the constant $\bar{\alpha}$ in (14), and thus the “more negative” is the time-derivative $\dot{V}(t_k^+)$ for $x_k \notin \mathcal{U}$ or $(x_k, \hat{\theta}_k) \notin \mathcal{S}_k$.

It has to be emphasized, though, that the arbitrary good transient performance mentioned above is made possible under the assumption that Δt can be made arbitrarily small. However, since in most applications there are practical limitations that forbid the choice of Δt beyond a lower bound, there will be practically an upper bound on α_k (or, equivalently, on $\bar{\alpha}$ in (14)) such that if we increase α_k beyond this upper bound we may end up with poor or even unstable performance. See [3] for more details on this issue.

6 Simulation Experiments

In order to test the efficiency of the proposed control scheme, we tested it by means of simulation experiments to the problem of image based visual servo control of an aerial robot vehicle. The dynamics of the overall system are described as follows [3]:

$$\begin{aligned}\dot{\delta} &= -\text{sk}(\Omega)\delta_1 - QV - \text{sk}(\Omega)q^* \\ M\dot{V} &= -M\text{sk}(\Omega)V + Te_3 + MgR^\tau e_3 \\ \dot{R} &= R\text{sk}(\Omega) \\ \mathbf{I}\dot{\Omega} &= -\text{sk}(\Omega)\mathbf{I}\Omega + [u_1, u_2, u_3]^\tau \\ \dot{T} &= u_4\end{aligned}\tag{15}$$

where $\delta, V, \Omega \in \mathbb{R}^3$ denote the image based error, translation velocity and angular velocity, respectively, T, M, g, \mathbf{I}, q^* denote the external force, mass, gravity coefficient, diagonal inertia matrix, and target vector (expressed in the camera-fixed frame), respectively, R is the rotational matrix, $Q \in \mathbb{R}^{3 \times 3}$ is a positive definite constant matrix, $e_3 = [0, 0, 1]^\tau$, $u = [u_1, u_2, u_3, u_4]^\tau$ denotes the control input vector and $\text{sk}(\Omega)$ denotes the skew-symmetric matrix such that $\text{sk}(\Omega)v = \Omega \times v$ for the vector product \times and any vector $v \in \mathbb{R}^3$. The interested reader is referred to [3] for more details on the above system. The system parameters and initial conditions assumed in the simulations were the same as the ones assumed in the simulations of [3], i.e. $M = 1\text{kg}$, $\mathbf{I} = \text{diag}(0.42, 0.42, 0.22)\text{kg m}$, $g = 9.8\text{ms}^{-1}$, $V(0) = \Omega(0) = 0$, $T(0) = g$, and $R(0) = I_3$. It has to be emphasized that the problem of constructing a controller for the system (15) is a quite challenging and difficult task even in the case where the system (15) dynamics are assumed known, see [3]. Note also that the system (15) contains $n = 19$ states and $m = 4$ control inputs.

The design for the proposed controller was as follows: the compact subset $\bar{\mathcal{X}}$ was set according to $\bar{\mathcal{X}} = \{x : \mathbb{R}^n : |x| \leq 50\}$ which can be seen that it includes the subset of all practically feasible states for the state vector x . For the construction of the monomial z_1 we choose, randomly, $L = 100$ monomial terms up to order three, resulting in a third order monomial with size L equal to 100. For the selection of the monomial z_2 , by observing that the input vector-field $g(x)$ for the particular system (15) corresponds to a constant matrix, we choose all of its L terms to be constant and equal to 1; it can be seen that such a choice does not cause any problem (especially with respect to the satisfaction of the PE conditions).

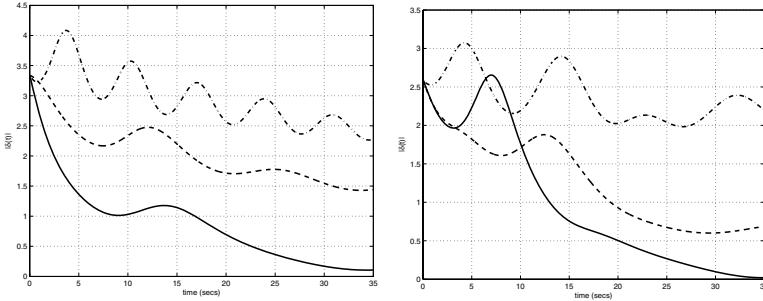


Fig. 1. Time history of $\delta(t)$ for the 1st (left subplot) and 2nd (right subplot) simulation experiments: the dash-dotted curve corresponds to $\beta_1 = 1$, the dashed curve corresponds to $\beta_1 = 20$ and the solid curve corresponds to $\beta_1 = 50$

We have used the above choices for $z_1(x)$ and $z_2(x)$ in the optimization problem (7) in order to on-line construct the CLF matrix $\mathbf{V}(\hat{\theta}_k)$. The number N of random points x_i was set equal to $N = L(L + m + 1) = 10500$; the algorithm of [9], appropriately adapted to be applicable to the optimization problem (7), was used for numerically solving the optimization problem (7). The user-defined constants ϵ_1, ϵ_2 were chosen according to $\epsilon_1 = \epsilon_2 = 0.5$. It has to be emphasized that different choices for the above constants had little effect on the efficiency of the overall control scheme. The rest of the controller design parameters and functions were chosen as follows: the constants a, β in (5) were set equal to 1 while the adaptive gain γ was chosen equal to 1 (for different choices of γ see remarks at the end of this section); the constants $\hat{\epsilon}_1, \hbar$ were chosen according to $\hat{\epsilon}_1 = 0.01, \hbar = 100/\gamma$ and $\alpha_k = \beta_2$ if $\left| \frac{\partial \mathcal{V}^\tau}{\partial x}(\hat{\theta}_k, x_k) \bar{g}(\hat{\theta}_k, x_k) \right| < \hat{\epsilon}_1$ or $k \leq \hbar$ and $\alpha_k = \beta_1$ otherwise, with $\beta_2 = 1$ and $\beta_1 \in \{1, 10, 50\}$. The sampling time Δt was chosen to be equal to 10^{-3} and the initial parameters estimates $\hat{\theta}(0)$ were set equal to 0 for all i . Such an initialization – although it is not likely to be adopted in practical situations since it corresponds to a non-stabilizable estimate for the stabilizable system (15) – was made in order to evaluate the performance of the proposed control scheme in cases where the estimated system dynamics correspond to a non-stabilizable system; we will comment on the importance of such an evaluation later in this section.

Two different sets of simulation experiments were created where, in each of these experiments, the entries of the vectors $\delta(0), q^*$ were randomly chosen in the interval $[-3, 3]^3$ and the matrix Q was a randomly-chosen positive definite matrix satisfying $\lambda(Q) \in (0, 2]$ where $\lambda(Q)$ denotes any of the eigenvalues of Q ; we have to emphasize that the subset of $\delta(0), q^*, Q$ generated as above is a superset of the corresponding set of $\delta(0), q^*, Q$ for which the control methodology of [3] guarantees closed-loop stability. Figure 1 exhibits the closed-loop system performance – in terms of the time-history of $\delta(t)$ – for the two aforementioned simulations experiments.

References

1. Berg, C., Christensen, J.P.R., Ressel, P.: Positive definite functions on Abelian groups. *Math. Ann.* 223, 253–274 (1976)
2. Ioannou, P.A., Sun, J.: *Stable and Robust Adaptive Control*. Prentice Hall, Englewood Cliffs (1995)
3. Hamel, T., Mahony, R.: Image based visual servo-control for a class of aerial robotic systems. *Automatica* 43, 1975–1983 (2007)
4. Kosmatopoulos, E.B., Polycarpou, M.M., Christodoulou, M.A., Ioannou, P.A.: High-Order Neural Network Structures for Identification of Dynamical Systems. *IEEE Transactions on Neural Networks* 6(2), 422–431 (1995)
5. Kosmatopoulos, E.B., Ioannou, P.A.: Robust switching adaptive control of multi-input nonlinear systems. *IEEE Transactions on Automatic Control* 47, 610–624 (2002)
6. Kosmatopoulos, E.B.: An adaptive optimization scheme with satisfactory transient performance. *Automatica* (in press); for a preliminary version, see also Kosmatopoulos, E.B., Papageorgiou, M.: An efficient adaptive optimization scheme. In: 17th IFAC World Congress 2008, Seoul, Korea, July 6-11, pp. 5071–5076 (2008)
7. Kosmatopoulos, E.B., Kouvelas, A.: Large-Scale Nonlinear Control System Fine-Tuning through Learning. *IEEE Transactions Neural Networks* (accepted under minor revision)
8. Kosmatopoulos, E.B.: Adaptive Control Design based on Adaptive Optimization Principles. *IEEE Transactions on Automatic Control* 53(11), 2680–2685 (2008)
9. Krislock, N., Lang, J., Varah, J., Pai, D.K., Seidel, H.-P.: Local compliance estimation via positive semidefinite constrained least squares. *IEEE Transactions on Robotics* 6, 1007–1011 (2004)
10. Lasserre, J.B.: A sum of squares approximation of nonnegative polynomials. *SIAM Review* 49(4), 651–669 (2007)
11. Maiorov, V., Meir, R.S.: Approximation bounds for smooth functions in $C(\Re^d)$ by neural networks and mixture networks. *IEEE Transactions on Neural Networks* 9, 969–978 (1998)
12. Nocedal, J., Wright, S.J.: *Numerical Optimization*. Springer, New York (2000)
13. Rubio, J.D., Yu, W.: Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm. *Neurocomputing* 70 (August 2007)
14. Sontag, E.D.: A universal construction of Arstein's theorem on nonlinear stabilization. *Systems & Control Letters* 13, 117–123 (1989)
15. Tsinias, J.: Sufficient Lyapunov-like conditions for stabilization. *Math. Contr., Sign., Syst.* 2, 343–357 (1989)

High-Order Fuzzy Switching Neural Networks: Application to the Tracking Control of a Class of Uncertain SISO Nonlinear Systems

Haris E. Psillakis

Department of Electronic & Computer Engineering
Technical University of Crete, Chania, 73100, Greece
psilakish@hotmail.com

Abstract. In this paper, a high-order neuro-fuzzy network (HONFN) with improved approximation capability w.r.t. the standard high-order neural network (HONN) is proposed. In order to reduce the overall approximation error, a decomposition of the neural network (NN) approximation space into overlapping sub-regions is created and different NN approximations for each sub-region are considered. To this end, *the HONFN implements a fuzzy switching among different HONNs* as its input vector switches along the different sub-regions of the approximation space. The HONFN is then used to design an adaptive controller for a class of uncertain single-input single-output nonlinear systems. The proposed scheme ensures the semiglobal uniform ultimate boundedness of the tracking error within a neighborhood of the origin and the boundedness of the NN weights and control law. Furthermore, a minimal HONFN, with two properly selected fuzzy rules, guarantees that the resulting ultimate bound does not depend on the unknown optimal approximation error (as is the case for classical adaptive NN control schemes) but solely from constants chosen by the designer. A simulation study is carried out to compare the proposed scheme with a classical HONN controller.

1 Introduction

Neuro-fuzzy networks (NFNs) have been under intensive research over the last years [1]-[12] since they utilize the advantages of both fuzzy logic and neural networks (NNs) i.e. the ability for human-like reasoning of fuzzy systems (in the form of IF-THEN rules) and the low-level learning capability of NNs. The NARA and FLNFN models introduced in [1] and [11] employ the NN in the consequent part of the fuzzy rules and can be used efficiently for control design yielding improved performance w.r.t. classical adaptive NN designs [13]-[18].

In classical adaptive NN control [13]-[18], NNs are employed for function approximation within some compact region (approximation region). Standard analysis proves that the tracking error depends (among others) on the optimal approximation error within that region. It is well known (see [13]-[18]), that the magnitude of that error depends on the number of neurons (fuzzy rules) that are used and the size of the approximation region. To improve tracking performance,

a suitable novel NFN is introduced in this paper and applied to the nonlinear control design problem. The motivating idea behind the proposed network is to create a decomposition of the classical NN approximation region into smaller sub-regions and consider different NN approximations for each sub-region in order to reduce the overall optimal approximation error. In this way, the optimal approximation error is reduced since the same number of neurons is used for a function approximation within some significantly smaller compact set. Obviously, in each of these regions, the ideal NN weight vectors that correspond to the optimal approximation (within the sub-region), are different in general. To take into account these different NN approximations in each of the sub-regions, the proposed NFN implements a fuzzy switching between NN weights that are estimates of the individual optimal NN weights of the currently active sub-region of the approximation space. If, then, this NFN (fuzzy switching NN) is used to control some nonlinear system, a reduced output tracking error will be obtained w.r.t. standard adaptive NN controllers as the overall optimal approximation error is reduced.

In the proposed scheme, high order neural networks (HONNs) [19], [15] are considered in order to keep the mathematical analysis as simple as possible (HONNs are linear-in-the-parameters networks) and to make possible the use of the same basis vector when switching from one sub-region to the other (to this end, a globally supported basis vector is needed). We note that other types of NNs with the above properties can also be used such as functional link networks resulting then to a functional link neuro-fuzzy network (FLNFN) [11].

Overlapping sub-regions are considered for the switching process to avoid discontinuous control action while suitable membership functions are introduced to account for the degree of residence of the NFN input vector within each sub-region. The result is a function approximation in the whole approximation space by a fuzzy-switching HONN (the HONFN) in the spirit of NARA networks [1].

An adaptive HONFN control design is then proposed for tracking control of SISO uncertain nonlinear systems in normal form. Practical stability (i.e. semiglobal uniform ultimate boundedness) of the tracking error and boundedness of the NN weights and control input is proved. Furthermore, a particular selection of only two fuzzy rules and suitable update laws results in an ultimate bound that does not depend on the unknown optimal approximation error (as is the case for standard adaptive NN control) but only on the design constants.

The following notation will be used throughout the paper: Ω' is the complement of the set Ω while $\partial\Omega$ defines the boundary of the set Ω . Also $\|\cdot\|$ denotes the standard Euclidean norm and \mathcal{L}_∞ is the space of all bounded continuous functions.

2 Fuzzy Switching HONN

2.1 Standard HONNs

HONNs are well known for their simple structure and their ability to approximate unknown smooth nonlinear functions. A HONN $W^T S(z)$ with input vector

$z \in \mathbf{R}^m$, weight vector $W \in \mathbf{R}^l$, node number l and basis function vector $S(z) \in \mathbf{R}^l$ can be used to approximate an unknown nonlinear function $F(z)$ within some compact set [19]. The vector of basis functions is $S(z) = [s_1(z) \ s_2(z) \ \cdots \ s_l(z)]^T$ where $s_i(z) = \prod_{j \in I_i} [\tanh(z_j)]^{d_j(i)}$, $i = 1, 2, \dots, l$, $z = [z_1 \ z_2 \ \cdots \ z_m]^T$; $\{I_1, I_2, \dots, I_l\}$ is a collection of l not ordered subsets of $\{1, 2, \dots, m\}$ and $d_j(i)$ are nonnegative integers. According to the universal approximation property of HONNs, if l is chosen sufficiently large, then $W^T S(z)$ can approximate any continuous function $F(z)$ to any desired accuracy over a compact set Ω_z [19]. Thus, for the smooth function $F(z)$ the following approximation holds true

$$F(z) = W^{*T} S(z) + \epsilon_a(z), \quad \forall z \in \Omega_z \quad (1)$$

with bounded function approximation error $|\epsilon_a(z)| \leq \epsilon_m$, $\forall z \in \Omega_z$ and ideal weight W^* defined as

$$W^* := \arg \min_{W \in \Omega_w} \left\{ \sup_{z \in \Omega_z} |F(z) - W^T S(z)| \right\} \quad (2)$$

where $\Omega_w := \{W \in \mathbf{R}^l : \|W\| \leq w_m\}$ with $w_m > 0$ chosen by the designer. The magnitude of the optimal approximation error ϵ_m depends on the choices of node number l and constrained set Ω_w . To reduce ϵ_m a larger number of nodes l and a larger value for the constrained set bound w_m should be considered.

2.2 Fuzzy Switching HONNs

Let us now decompose the approximation region Ω_z into different overlapping approximation sub-regions Ω_{zi} ($i = 1, 2, \dots, q$) such that $\Omega_z = \bigcup_{i=1}^q \Omega_{zi}$ and consider different standard HONN approximations [19] for each sub-region Ω_{zi} ($i = 1, 2, \dots, q$) i.e.

$$F(z) = W_i^{*T} S(z) + \epsilon_{ai}(z), \quad \forall z \in \Omega_{zi} \quad (i = 1, 2, \dots, q) \quad (3)$$

with bounded function approximation error $|\epsilon_{ai}| \leq \epsilon_{mi}$, $\forall z \in \Omega_{zi}$ and ideal weights W_i^* defined as

$$W_i^* := \arg \min_{W \in \Omega_w} \left\{ \sup_{z \in \Omega_{zi}} |F(z) - W^T S(z)| \right\} \quad (i = 1, 2, \dots, q). \quad (4)$$

Then, the optimal approximation error ϵ_{mi} in each sub-region is less or equal than the optimal approximation error ϵ_m in the entire region, i.e. $\epsilon_{mi} \leq \epsilon_m$, ($i = 1, 2, \dots, q$), since the property $\Omega_{zi} \subset \Omega_z$ yields

$$\sup_{z \in \Omega_{zi}} |F(z) - W^T S(z)| \leq \sup_{z \in \Omega_z} |F(z) - W^T S(z)|, \quad \forall W \in \Omega_w, \quad (i = 1, 2, \dots, q).$$

As $\epsilon_{mi} \leq \epsilon_m$ holds true $\forall i = 1, \dots, q$ we respectively have that $\max_{1 \leq i \leq q} \{\epsilon_{mi}\} \leq \epsilon_m$. Thus, a decomposition of the approximation region into overlapping sub-regions wherein different NN approximations are considered can actually reduce the overall optimal approximation error. The function approximation described

above that employs different optimal NN weights in different (overlapping) sub-regions of the approximation region allows for a fuzzy switching-NN approximation in the following sense:

Define the membership functions $m_i : \Omega_z \rightarrow [0, 1]$, ($i = 1, 2, \dots, q$) given by

$$m_i(z) := \begin{cases} 1 & z \in \Omega_{zi} \setminus \bigcup_{j=1, j \neq i}^q \Omega_{zj} := \Omega_{i1} \\ h_i(z) & z \in \Omega_{zi} \cap \bigcup_{j=1, j \neq i}^q \Omega_{zj} := \Omega_{i2} \\ 0 & z \in \Omega_z \setminus \Omega_{zi} := \Omega_{i3} \end{cases} \quad (5)$$

where $h_i : \Omega_{i2} \rightarrow [0, 1]$ are arbitrary continuous functions with $\lim_{z \rightarrow z_0} h_i(z) = 1$ $\forall z_0 \in \partial\Omega_{i1} \cap \partial\Omega_{i2}$, $\lim_{z \rightarrow z_1} h_i(z) = 0 \forall z_1 \in \partial\Omega_{i2} \cap \partial\Omega_{i3}$ and $\sum_{i=1}^m h_{j_i}(z) = 1$, $\forall z \in (\bigcap_{i=1}^m \Omega_{zj_i}) \cap (\bigcap_{k=m+1}^q \Omega'_{zj_k})$ where the index set $J = \{j_1, j_2, \dots, j_m, j_{m+1}, \dots, j_q\}$ is just a rearrangement of $\{1, 2, \dots, q\}$ (i.e. $\sum_{i=1}^m h_{j_i}(z) = 1$ whenever z belongs to the area wherein the regions $\Omega_{zj_1}, \dots, \Omega_{zj_m}$ overlap while $\Omega_{zj_{m+1}}, \dots, \Omega_{zj_q}$ do not). In this way m_i represents a well-defined continuous indicator of the membership of z in Ω_{zi} and $\sum_{i=1}^q m_i(z) = 1, \forall z \in \Omega_z$.

Introducing therefore the membership functions m_i of (5), then, the following *neuro-fuzzy approximation property* holds true for the smooth function $F(z)$

$$F(z) = \sum_{k=1}^q m_k(z) [W_k^{*T} S(z) + \bar{\epsilon}_{ak}(z)], \forall z \in \Omega_z \quad (6)$$

where W_i^* is defined from (4) and $\bar{\epsilon}_{ai}(z)$ is an extension of $\epsilon_{ai}(z)$ of (3) to Ω_z defined as

$$\bar{\epsilon}_{ai}(z) := \begin{cases} \epsilon_{ai}(z) & z \in \Omega_{zi} \\ 0 & z \in \Omega_z \setminus \Omega_{zi} \end{cases} \quad (7)$$

To prove (6) we must consider the following two cases: i) $z \in \Omega_{i1} \subset \Omega_{zi}$. Then $m_i(z) = 1$ and $m_j(z) = 0, \forall j \neq i$ and (3) yields the desired result. ii) $z \in \Omega_{i2}$ i.e. there exists more than one regions $\Omega_{zj_1}, \Omega_{zj_2}, \dots, \Omega_{zj_m}$ (including Ω_{zi}) with $j_k \in \{1, 2, \dots, q\}$ such that $z \in (\bigcap_{i=1}^m \Omega_{zj_i}) \cap (\bigcap_{k=m+1}^q \Omega'_{zj_k})$. Then

$$\begin{aligned} F(z) &= \sum_{i=1}^q m_i(z) F(z) = \sum_{i=1}^m h_{j_i}(z) F(z) \\ &= \sum_{i=1}^m h_{j_i}(z) [W_{j_i}^{*T} S(z) + \epsilon_{aj_i}(z)] = \sum_{i=1}^q m_i(z) [W_i^{*T} S(z) + \bar{\epsilon}_{ai}(z)] \end{aligned} \quad (8)$$

as $m_i(z) = 0$ for all $i \notin \{j_1, j_2, \dots, j_m\}$ that completes the proof of (6). From the approximation (6) we obtain

$$\left| F(z) - \sum_{i=1}^q m_i(z) W_i^{*T} S(z) \right| \leq \sum_{i=1}^q m_i(z) |\bar{\epsilon}_{ai}(z)| \leq \max_{1 \leq i \leq q} \{\epsilon_{mi}\} \quad (9)$$

that yields the desired property i.e. the optimal approximation error within some compact region using the HONFN $\sum_{i=1}^q m_i(z) W_i^{*T} S(z)$ is less or equal than the maximum of the optimal approximation error of an individual HONN within

the different subregions. The particular selection of fuzzy rules and the resulting NFN can be considered as an implementation of the following fuzzy switching logic scheme

IF $z \in \Omega_{z1}$ THEN use the HONN $W_1^T S(z)$

\vdots

IF $z \in \Omega_{zq}$ THEN use the HONN $W_q^T S(z)$

i.e. the network fuzzy switches between HONNs with different weights. To emphasize this property we name this NFN the *fuzzy switching HONN*.

Remark 1. The fuzzy switched HONN is similar to the NARA model [1] and the FLNFN introduced in [2] wherein multilayer and functional link NNs have been employed for the consequent part of the network. Furthermore, the previous neuro-fuzzy approximation property and the accompanying analysis can be used to explain their improved approximation capability [1], [2] over standard NNs.

In the next, our main focus is the application of the fuzzy switched HONNs to the tracking control problem.

3 Plant Description and Control Problem Formulation

Consider the following class of SISO systems in normal form

$$\begin{aligned}\dot{x}_i &= x_{i+1}, \quad i = 1, 2, \dots, n-1 \\ \dot{x}_n &= a(x) + b(x)u, \quad y = x_1\end{aligned}\tag{10}$$

where $x := [x_1, x_2, \dots, x_n]^T$ is the state vector and $u \in \mathbf{R}$, $y \in \mathbf{R}$ the control input and the system output, respectively. The functions $a : \mathbf{R}^n \rightarrow \mathbf{R}$, $b : \mathbf{R}^n \rightarrow \mathbf{R}$ represent the system's unknown smooth nonlinearities. The control objective is the output $y(t)$ to follow closely (track) some desired reference signal $y_d(t)$.

Assumption 1. The sign of $b(x)$ is known and there exist constants $b_0 > 0$ (unknown), $\delta_1 > 0$ (known) and a known class \mathcal{K} function (i.e. zero at zero and strictly increasing) $\hat{\gamma}_1(\cdot)$ such that $b_0 \leq |b(x)| \leq \hat{\gamma}_1(\|x\|) + \delta_1 := b_1(x)$, $\forall x \in \mathbf{R}^n$.

Remark 2. Assumption 1 poses a controllability condition on system (10) ([7], [8]) since it implies that the smooth function $b(x)$ is either strictly positive or strictly negative. Without loss of generality, we assume from now on that $b(x) > 0$, $\forall x \in \mathbf{R}^n$.

Define now the vectors $x_d := [y_d, \dot{y}_d, \dots, y_d^{(n-1)}]^T$, $e := x - x_d = [e_1, e_2, \dots, e_n]^T$ and the filtered tracking error $s := (d/dt + \lambda)^{n-1} e_1 = [\Lambda^T 1]^T e$ with $\lambda > 0$ and $\Lambda := [\lambda^{n-1}, (n-1)\lambda^{n-2}, \dots, (n-1)\lambda]^T$. Then, the time derivative of s can be written as

$$\dot{s} = a(x) + b(x)u - v\tag{11}$$

with $v := y_d^{(n)} - [\Lambda^T 1]^T e$ a known time-varying signal.

Assumption 2. The reference signal y_d and its derivatives $\dot{y}_d, \ddot{y}_d, \dots, y_d^{(n)}$ are all known, smooth and bounded i.e. there exist known $r_i \geq 0$ such that $|y_d^{(i)}| \leq r_i$, $i = 0, 1, \dots, n$.

4 Control Design

4.1 Adaptive NFN Controller

For system (10) the following adaptive HONFN controller is considered in this paper

$$u(t) = \frac{1}{b_1(x)} \left[-k(t)s - \sum_{i=1}^q m_i(z) \hat{W}_i^T S(z) \right] \quad (12)$$

with $k(t) := k_0 + k_1 [\hat{\gamma}_1 (\|\bar{x}_{n-1}\| + |s| + |v_1|) + \delta_1]$, $k_0, k_1 > 0$, $\bar{x}_{n-1} := [x_1, \dots, x_{n-1}]^T$, $v_1 := y_d^{(n-1)} - [A^T \ 0]^T e$ and $z := [x^T, s, |v|]^T$ the input to the NFN given by $\sum_{i=1}^q m_i(z) \hat{W}_i^T S(z)$ that evolves within some compact approximation region $\Omega_z := \{z : |s| \leq C_0, |y_d^{(i)}| \leq r_i, i = 0, 1, 2, \dots, n\} \subset \mathbf{R}^{n+2}$ (C_0 is some constant that depends on the initial conditions). The NFN implements a decomposition of this compact set Ω_z into arbitrary, chosen by the designer, overlapping sub-regions Ω_{zi} ($i = 1, 2, \dots, q$) with membership functions $m_i(z)$ (defined as in (5)), $i = 1, 2, \dots, q$ that fuzzy switch between the q HONNs $\hat{W}_i^T S(z)$ ($i = 1, 2, \dots, q$); \hat{W}_i is the estimate of W_i^* that gives the optimal approximation of some nonlinear function $F(z)$ within Ω_{zi} .

The following NN weight adaptation laws with projection and deadzone are selected

$$\begin{aligned} \dot{\hat{W}}_i &= \chi_{\{|s|>\eta\}}(t) \text{Proj} \left\{ \gamma_i m_i(z) S(z) s \right\} \\ &= \begin{cases} \chi_{\{|s|>\eta\}}(t) \gamma_i m_i(z) S(z) s & \text{if } \|\hat{W}_i\| < w_m \text{ or } \|\hat{W}_i\| = w_m \\ & \text{and } m_i(z) \hat{W}_i^T S(z) s \leq 0 \\ \gamma_i m_i(z) \chi_{\{|s|>\eta\}}(t) s \left[S(z) - \frac{\hat{W}_i^T S(z)}{\|\hat{W}_i\|^2} \hat{W}_i \right] & \text{if } \|\hat{W}_i\| = w_m \\ & \text{and } m_i(z) \hat{W}_i^T S(z) s > 0 \end{cases} \end{aligned} \quad (13)$$

where the positive constants $\gamma_i > 0$ are the adaptation gains and $\chi_{\{|s|>\eta\}}(t)$ is the characteristic function of the set $\{s : |s| > \eta\}$ defined as $\chi_{\{|s|>\eta\}}(t) = 1$ whenever $|s(t)| > \eta$ and $\chi_{\{|s|>\eta\}}(t) = 0$ when $|s(t)| \leq \eta$ where $\eta > 0$ is some small design constant. This learning algorithm guarantees that $\|\hat{W}_i(t)\| \leq w_m$, $\forall t \geq 0$ if initially $\|\hat{W}_i(0)\| \leq w_m$ (see [15], [18]). For the proposed controller a Lyapunov stability analysis using the continuous nonnegative integral function

$$V_s := \begin{cases} \int_{\eta \operatorname{sgn}(s)}^s \beta_0(\bar{x}_{n-1}, \sigma + v_1) \sigma d\sigma & \text{if } |s| > \eta \\ 0 & \text{if } |s| \leq \eta \end{cases} \quad (14)$$

with $\beta_0(x) := b_1(x)/b(x)$ can be used to prove the following Theorem (the proof is omitted due to space limitations).

Theorem 1. Let the nonlinear system (10) and the reference trajectory y_d satisfying Assumptions 1-2. Then, for the adaptive HONFN control law (12) with update laws (13) and membership functions (5) defined on some overlapping sub-regions of a sufficiently large approximation space $\Omega_z := \{z : |s| \leq C_0, |y_d^{(i)}| \leq r_i, i = 0, 1, 2, \dots, n\} \subset \mathbf{R}^{n+2}$ (where $C_0 \geq \sqrt{2}[V_s(0) + \lambda_0]^{1/2} + \eta$ with $\lambda_0 := (1/8b_0k_0k_1)(\max_{1 \leq i \leq q}\{\epsilon_{mi}\} + 2w_m s_m)^2$, $\|S(z)\| \leq s_m$) it holds true that i) all the closed-loop signals remain bounded and the tracking error enters in finite time within the region $\Omega_{1\epsilon} := \{e_1 : |e_1| \leq (1/\lambda^{n-1})(\sqrt{2\lambda_0} + \eta + \epsilon)\}$ for any (arbitrarily small) $\epsilon > 0$ wherein it remains thereafter and ii) the mean square filtered tracking error is bounded from

$$\lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t s^2(\tau) d\tau \leq \max \left\{ \frac{\max_{1 \leq i \leq q} \epsilon_{mi}^2}{(k_0 + k_1 \delta_1)^2}, \eta^2 \right\} \quad (15)$$

for a sufficiently large chosen projection barrier ($w_m \geq \max_{1 \leq i \leq q} \|W_i^*\|$).

Remark 3. As can be seen from (15) the smaller the deadzone is selected, the smaller the mean square filtered tracking error is. Thus, it is reasonable for someone to consider completely removing the deadzone ($\eta = 0$). However, the deadzone's utility will be proven in the next Section, where it is shown that (for the proposed scheme) a proper selection of two fuzzy rules only ensures asymptotic convergence of the filtered tracking error s to $\{s : |s| \leq \eta\}$ despite the number of nodes used in the HONNs (i.e. excellent tracking can be achieved with relatively few nodes).

4.2 Minimal Implementation for Desired Tracking Performance

Let us define a special form of the NFN in the control law (12), with two fuzzy rules based on the filtered tracking error s . Particularly, the selection of the two membership functions

$$m_1(z) = \min\{1, \max\{0, (1/2\eta)(s + \eta)\}\}, \quad m_2(z) = 1 - m_1(z), \quad \forall z \in \Omega_z \quad (16)$$

ensures asymptotic convergence of s to the interval $[-\eta, \eta]$ which in turn implies asymptotic convergence of the tracking error $y - y_d$ to $[-\eta/\lambda^{n-1}, \eta/\lambda^{n-1}]$ (due to space limitations the proof is also omitted).

Theorem 2. Let the SISO uncertain nonlinear system (10) satisfying Assumption 1 and the desired reference trajectory y_d for which Assumption 2 holds true. Then, the selection of the adaptive HONFN control law (12) with two fuzzy rules only ($q = 2$) defined by the membership functions (16) and the NN weight adaptation laws with projection and deadzone (13) (for which a sufficiently large projection barrier has been chosen), ensures boundedness of all the closed-loop variables and asymptotic convergence of the tracking error $y - y_d$ to the interval $[-\eta/\lambda^{n-1}, \eta/\lambda^{n-1}]$.

Remark 4. Theorem 2 gives the ultimate bound η/λ^{n-1} on the tracking error magnitude that depends solely on constants chosen by the designer and can

be reduced at will. This improves the result of Section 4.1 and other standard results on classical NN adaptive control that yield bounds on the tracking error which depend on the NN optimal approximation error [13]-[18].

5 Simulation Study

To illustrate the effectiveness of the proposed approach, let now a Van der Pol oscillator driven by a nonlinear control input with dynamics described from (10) with $n = 2$, $a(x) = -x_1 + (1 - x_1^2)x_2$, $b(x) = 1 + x_2^2$, initial conditions $x(0) = [1 \ 0]^T$ and reference signal $y_d(t) = \sin t$. The adaptive HONFN control law of Theorem 2 is implemented with $\hat{\gamma}_1(s) = s^2$, $\delta_1 = 1$, $k_0 = k_1 = 0.1$, $\lambda = 1$, $\eta = 2 \times 10^{-3}$ and 2 second order NNs with 4 inputs and 15 nodes each. For the update laws (13) the gains $\gamma_1 = \gamma_2 = 1/2$ and the weight norm bound $w_m = 10$ are chosen. The controller's performance is compared to a HONN adaptive controller without fuzzy switching of the form

$$u(t) = \frac{1}{b_1(x)} [-k(t)s - \hat{W}^T S(z)] \quad (17)$$

with $k(t)$ defined as before and the projection-based adaptation law

$$\dot{\hat{W}} = \text{Proj}\{\gamma S(z)s\}. \quad (18)$$

The effectiveness of the controller (17) with the NN weight update law (18) to the tracking control problem can be proved using standard arguments. In the simulation study, two cases are considered for controller (17): i) a second order NN with 15 nodes and ii) a third order NN with 31 nodes (more nodes than the total number of nodes in the fuzzy-switching HONN). The design parameters of the HONFN are also used for this scheme. Simulation results (Fig. 1-2) indicate that both controllers provide effective tracking of the desired reference signal with bounded NN weights and control signals. However, the adaptive HONFN controller yields improved tracking performance w.r.t. the classical HONN control law even for the case ii) where more NN nodes are considered in the classical scheme. The convergence of s to the region $\{s : |s| \leq \eta\}$ is clearly illustrated in Fig. 3.

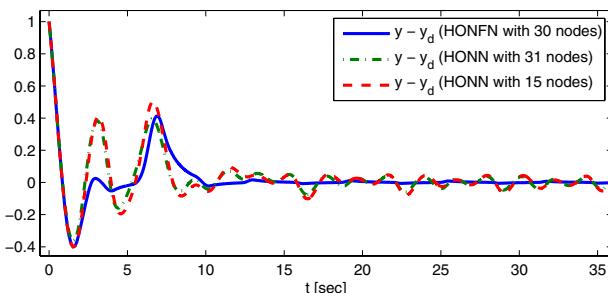
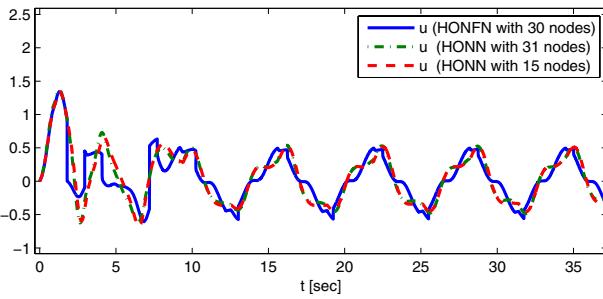
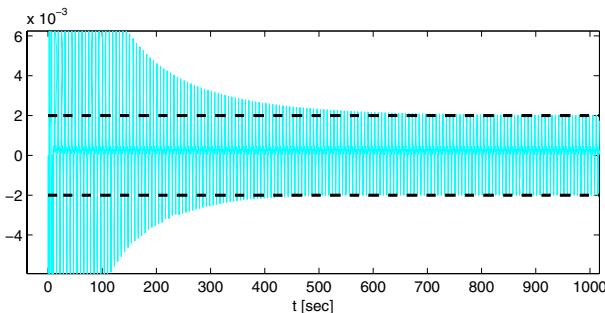


Fig. 1. The tracking error

**Fig. 2.** The control input**Fig. 3.** The convergence of s in $\{s : |s| \leq \eta\}$

6 Conclusion

Fuzzy switching HONNs are introduced in this paper and effectively applied to the tracking control problem of uncertain nonlinear systems in normal form. A particular selection of fuzzy rules is proposed that yields an ultimate bound for the tracking error independent from the optimal NN approximation error.

References

1. Takagi, H., Suzuki, N., Koda, T., Kojima, Y.: Neural networks designed on approximate reasoning architecture and their application. *IEEE Trans. Neural Networks* 3, 752–759 (1992)
2. Jang, J.-S.R.: ANFIS: Adaptive network-based fuzzy inference system. *IEEE Trans. Syst., Man & Cybern.* 23, 665–685 (1993)
3. Lin, C.T., Lee, C.S.G.: *Neural-Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice-Hall, NJ (1996)
4. Lin, C.J., Lin, C.T.: An ART-based fuzzy adaptive learning control network. *IEEE Trans. Fuzzy Systems* 5(4), 477–496 (1997)
5. Juang, C.F., Lin, C.T.: An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Trans. Fuzzy Systems* 6(1), 12–31 (1998)

6. Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: survey in soft computing framework. *IEEE Trans. Neural Networks* 11(3), 748–768 (2000)
7. Li, C., Lee, C.Y.: Self-organizing neuro-fuzzy system for control of unknown plants. *IEEE Trans. Fuzzy Systems* 11(1), 135–150 (2003)
8. Sun, F., Sun, Z., Li, L., Li, H.X.: Neuro-fuzzy adaptive control based on dynamic inversion for robotic manipulators. *Fuzzy Sets and Systems* 134, 117–133 (2003)
9. Da, F.P., Song, W.Z.: Fuzzy neural networks for direct adaptive control. *IEEE Trans. Ind. Electron.* 50(3), 507–513 (2003)
10. Liu, X.J., Lara-Rosano, F., Chan, C.W.: Model-reference adaptive control based on neurofuzzy networks. *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.* 34(3), 302–309 (2004)
11. Chen, C.-H., Lin, C.J., Lin, C.T.: A functional-link-based neuro-fuzzy network for nonlinear system control. *IEEE Trans. Fuzzy Syst.* 16, 1362–1378 (2008)
12. Boutalis, Y., Theodoridis, D.C., Christodoulou, M.A.: A new neuro-FDS definition for indirect adaptive control of unknown nonlinear systems using a method of parameter hopping. *IEEE Trans. Neural Netw.* 20(4), 609–625 (2009)
13. Ge, S.S., Lee, T.H., Harris, C.: Adaptive Neural Network Control of Robotic Manipulators. World Scientific, London (1998)
14. Lewis, F.L., Jagannathan, S., Yesildirek, A.: Neural Network Control of Robot Manipulators and Nonlinear Systems. Taylor and Francis, London (1999)
15. Rovithakis, G.A., Christodoulou, M.A.: Adaptive Control with Recurrent High-Order Neural Networks. Springer, London (2000)
16. Spooner, J.T., Maggiore, M., Ordóñez, R., Passino, K.M.: Stable Adaptive Control and Estimation for Nonlinear Systems- Neural and Fuzzy Approximator Techniques. Wiley, New York (2002)
17. Ge, S.S., Hang, C.C., Lee, T.H., Zhang, T.: Stable Adaptive Neural Network Control. Kluwer, London (2002)
18. Farrell, J.A., Polycarpou, M.M.: Adaptive Approximation Based Control: Unifying, Neural, Fuzzy and Traditional Adaptive Approximation Approaches. Wiley, New York (2006)
19. Kosmatopoulos, E.B., Polycarpou, M.M., Christodoulou, M.A., Ioannou, P.A.: High-order neural network structures for identification of dynamical systems. *IEEE Trans. Neural Netw.* 6, 422–431 (1995)

A Guide for the Upper Bound on the Number of Continuous-Valued Hidden Nodes of a Feed-Forward Network

Rua-Huan Tsaih^{1,*} and Yat-wah Wan²

¹ Department of Management Information Systems,
National Chengchi University, Taipei, Taiwan 116
tsaih@mis.nccu.edu.tw

² Graduate Institute of Global Operations Strategy and Logistics Management,
National Dong Hwa University, Taiwan 974
ywan@mail.ndhu.edu.tw

Abstract. This study proposes and validates a construction concept for the realization of a real-valued single-hidden layer feed-forward neural network (SLFN) with continuous-valued hidden nodes for arbitrary mapping problems. The proposed construction concept says that for a specific application problem, the upper bound on the number of used hidden nodes depends on the characteristic of adopted SLFN and the observed properties of collected data samples. A positive validation result is obtained from the experiment of applying the construction concept to the m -bit parity problem learned by constructing two types of SLFN network solutions.

Keywords: Bound, hidden nodes, single-hidden layer feed-forward neural network, preimage, parity problem.

1 Bound on the Number of Hidden Nodes

With regard to the realization of a real-valued single-hidden layer feed-forward neural network (SLFN) with continuous-valued hidden nodes for arbitrary mapping problems, this study proposes and validates the *construction concept*. The proposed construction concept says that for any learning problem with specific data relationship observed among input vectors and target values, knowledge of the characteristic of adopted SLFN and the observed data properties helps find a better upper bound on the number of used hidden nodes of network solution than the one obtained from the conventional construction method that misses the characteristic and ignores the relationship.

The question of the necessary number of hidden nodes for a feed-forward neural network has been addressed in [1][3][5][6]. [3] argued that fewer hidden nodes is generally regarded as desirable for preventing over-learning, but the necessary number of hidden nodes is not known in general. Both of [1] and [6]

* Corresponding author.

obtained a bound of $N-1$ for N distinct samples, but both assumed that the hidden layer of nodes produced a binary-valued output. To obtain the training advantages of back propagation of errors, most models use continuous-valued hidden node outputs, as we do here. The methods of [1] and [6] do not seem to generalize to our situation.

[5] adopted standard SLFNs with any nonlinear, continuous-valued activation function that has a limit at each infinity and claimed that N distinct samples can be fit perfectly through a SLFN with N hidden nodes. However, the number N is a loose upper bound on the number of hidden nodes of a SLFN solution for N distinct samples; for instance, [12] and [13] stated that the m -bit parity problem is solvable by a SLFN with merely $\lceil (m+1)/2 \rceil$ hidden nodes and with the sigmoid activation functions at hidden nodes, in which, and hereafter, $\lceil x \rceil$ denotes the smallest integer which is larger than or equal to x . In fact, instead of a loose and universal upper bound applied to the number of hidden nodes of SLFN solutions for all learning problems, most researchers and practitioners desire a concept to help find a better upper bound on the number of used hidden nodes of SLFN solution for a specific learning problem, as we do here. The discussion of [5][12][13] does not seem to provide such a concept. To address such challenge, we propose the construction concept.

In Section 2, this study shows that the conventional construction method of [5] misses the characteristic of the adopted SLFN and ignores the data relationship among input vectors and target values. This study then explores characteristics of SLFN through the preimage analysis, in which the preimage of a given output is the collection of inputs for the output. In Section 4, we set up the experiment of parity problem to validate the construction concept. The parity problem is a challenging benchmark for testing neural network learning algorithm. Some, but not exhaustive, recent studies of the parity problem can be found in [2][4][7][8][9][12][15]. Conclusions and future work are presented at the end.

2 A Conventional Construction Method

List of notations used in mathematical representations: Characters in bold represent column vectors, matrices or sets; $(\cdot)^T$ denotes the transpose of (\cdot) .

N \equiv the amount of training samples;

I \equiv the amount of input nodes;

J \equiv the amount of hidden nodes;

\mathbf{x} $\equiv (x_1, x_2, \dots, x_I)^T$: the input vector, in which x_i is the i^{th} input component, with i from 1 to I ;

\mathbf{a} $\equiv (a_1, a_2, \dots, a_J)^T$: the hidden activation vector, in which a_j is the activation value of the j^{th} hidden node, with j from 1 to J ;

y \equiv the activation value of the output node and $y = f(\mathbf{x})$ with f being the map function of \mathbf{x} and y ;

w_{ji}^H \equiv the weight between the i^{th} input variable and the j^{th} hidden node, in which the superscript H throughout the paper refers to quantities related to the hidden layer;

- $\mathbf{w}_j^H \equiv (w_{j1}^H, w_{j2}^H, \dots, w_{jI}^H)^T$, the vector of weights between the input variables and the j^{th} hidden node;
- $\mathbf{W}^H \equiv (\mathbf{w}_1^H, \mathbf{w}_2^H, \dots, \mathbf{w}_J^H)^T$, the $J \times I$ matrix of weights between the input variables and the hidden nodes;
- $w_{j0}^H \equiv$ the bias value of the j^{th} hidden node;
- $w_j^O \equiv$ the weight between the j^{th} hidden node and the output node in which the superscript O throughout the paper refers to quantities related to the output layer;
- $\mathbf{w}^O \equiv (w_1^O, w_2^O, \dots, w_J^O)^T$; and
- $w_0^O \equiv$ the bias value of the output node.

The construction method of [5] works for *any* activation function g as long as $g(x_{01}) \neq \lim_{x \rightarrow +\infty} g(x)$. Let \mathbf{x}^c and t^c be the c^{th} input pattern and the corresponding target value, respectively, $c = 1, \dots, N$. Without the loss of generality, assume that $\mathbf{x}^c \neq \mathbf{x}^d$ for $1 \leq c \neq d \leq N$. Let $\mathbf{T} \equiv (t^1, t^2, \dots, t^N)^T$ be the N -dimensional vector of target values for the N input samples; $x_{01} > x_{02}$ be two arbitrary pre-specified constants. The construction method first arbitrarily chooses an I -dimensional vector \mathbf{w} such that

$$\mathbf{w}^T \mathbf{x}^1 < \mathbf{w}^T \mathbf{x}^2 < \dots < \mathbf{w}^T \mathbf{x}^N. \quad (1)$$

For this \mathbf{w} , the construction method then calculates \mathbf{w}_j^H and w_{j0}^H from eqt. (B), in which the values of \mathbf{w}_j^H and w_{j0}^H are independent of the target outputs $\{t^c\}$:

$$\mathbf{w}_j^H = \begin{cases} \mathbf{0}, & \text{if } j = 1; \\ \frac{x_{02} - x_{01}}{\mathbf{w}^T \mathbf{x}^j - \mathbf{w}^T \mathbf{x}^{j-1}} \mathbf{w}, & \text{if } 2 \leq j \leq N; \end{cases} \quad (2)$$

$$w_{j0}^H = \begin{cases} x_{02}, & \text{if } j = 1; \\ \frac{x_{01} \mathbf{w}^T \mathbf{x}^j - x_{02} \mathbf{w}^T \mathbf{x}^{j-1}}{\mathbf{w}^T \mathbf{x}^j - \mathbf{w}^T \mathbf{x}^{j-1}}, & \text{if } 2 \leq j \leq N; \end{cases} \quad (3)$$

Let a_j^c be the j^{th} *activation value* for the c^{th} input, i.e., the output of the j^{th} hidden node for input \mathbf{x}^c . Then $a_1^c \equiv g(x_{02})$ and $a_j^c \equiv g(\frac{x_{02} - x_{01}}{\mathbf{w}^T \mathbf{x}^j - \mathbf{w}^T \mathbf{x}^{j-1}} \mathbf{w}^T \mathbf{x}^c + \frac{x_{01} \mathbf{w}^T \mathbf{x}^j - x_{02} \mathbf{w}^T \mathbf{x}^{j-1}}{\mathbf{w}^T \mathbf{x}^j - \mathbf{w}^T \mathbf{x}^{j-1}} \mathbf{w}) \forall 2 \leq j \leq N$. Let $\mathbf{a}^c \equiv (g(x_{02}), a_2^c, \dots, a_N^c)^T$ and $\mathbf{M} \equiv (\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^N)^T$. [5] showed that the N samples in $\{\mathbf{x}\}$ space are mapped to N distinctive points in the activation space such that the $N \times N$ matrix \mathbf{M} is invertible. With w_0^O set to zero, $\mathbf{w}^O = \mathbf{M}^{-1} \mathbf{T}$ can always be found to match $w_0^O + \sum_{j=1}^N w_j^O a_j^c$ to t^c without any error.

[5] ends up at the construction method with neither discussion on the characteristic of the adopted SLFN nor on the data relationship among input vectors and target values. For instance, the output value y of the constructed SLFN for an arbitrary input \mathbf{x} can be represented as $w_1^O g(x_{02}) + \sum_{j=2}^N w_j^O g(w_{j0}^H + \sum_{i=1}^I w_{j0}^H x_i)$, since a_1 always equals $g(x_{02})$. Thus $w_1^O g(x_{02})$ can serve as the bias of the output node such that there are only $N - 1$ effective hidden nodes. Furthermore, from eqt. (B), vectors \mathbf{w}_j^H for $2 \leq j \leq N$ are linearly dependent. Therefore, the constructed SLFN has the weight vectors (from the input layer) of all its (effective) hidden nodes linearly dependent on each other.

3 Characteristics of SLFN

We apply the following preimage analysis to explore characteristics of the SLFN. Without any loss of generality, assume the \tanh activation function is adopted in all hidden nodes.

Denote a particular collection of w_{j0}^H , \mathbf{w}_j^H , \mathbf{w}^O , and w_0^O by Θ . Given Θ , the mapping f of SLFN is the composite of the following mappings: the *activation mapping* $\Phi_A : \Re^I \rightarrow (-1, 1)^J$ that maps an input \mathbf{x} to an activation value \mathbf{a} (i.e., $\mathbf{a} = \Phi_A(\mathbf{x})$); and the *output mapping* $\Phi_O : (-1, 1)^J \rightarrow (w_0^O - \sum_{j=1}^J |w_j^O|, w_0^O + \sum_{j=1}^J |w_j^O|)$ that maps an activation value \mathbf{a} to an output y (i.e., $y = \Phi_O(\mathbf{a})$). Note that, since the range of Φ_A and the domain of Φ_O are set as $(-1, 1)^J$, the range in the output space $\Im \equiv (w_0^O - \sum_{j=1}^J |w_j^O|, w_0^O + |\sum_{j=1}^J w_j^O|)$ contains all achievable output values. For ease of reference in later discussion, we also call \Re^I the *input space* and $(-1, 1)^J$ the *activation space*.

Thus, $f^{-1}(y) \equiv \Phi_A^{-1} \circ \Phi_O^{-1}(y)$, with

$$\Phi_O^{-1}(y) \equiv \{\mathbf{a} \in (-1, 1)^J \mid \sum_{j=1}^J w_j^O a_j = y - w_0^O\}, \quad (4)$$

$$\Phi_A^{-1}(\mathbf{a}) \equiv \cap_{j=1}^J \{\mathbf{x} \in \Re^I \mid \sum_{i=1}^I w_{ji}^H x_i = \tanh^{-1}(a_j) - w_{j0}^H\}, \quad (5)$$

where $\tanh^{-1}(x) = 0.5 \ln(\frac{1+x}{1-x})$. Formally, the followings are defined for every given Θ :

- (a) A value $y \in \Re$ is *void* if $y \neq f(\{\Re^I\})$, i.e., for all $\mathbf{x} \in \Re^I$, $f(\mathbf{x}) \neq y$. Otherwise, y is *non-void*.
- (b) A point $\mathbf{a} \in (-1, 1)^J$ is *void* if $\mathbf{a} \notin \Phi_A(\Re^I)$, i.e., for all $\mathbf{x} \in \Re^I$, $\Phi_A(\mathbf{x}) \neq \mathbf{a}$. Otherwise, \mathbf{a} is *non-void*. The set of all non-void \mathbf{a} 's in the activation space is named as the *non-void set*.
- (c) The *image* of an input $\mathbf{x} \in \Re^I$ is $y \equiv f(\mathbf{x})$ for $y \in \Im$.
- (d) The *preimage* of a non-void output value y is $f^{-1}(y) \equiv \{\mathbf{x} \in \Re^I \mid f(\mathbf{x}) = y\}$. The preimage of a void value y is the empty set.
- (e) The *internal-preimage* of a non-void output value y is the collection $\{\mathbf{a} \in (-1, 1)^J \mid \Phi_O(\mathbf{a}) = y\}$ on the activation space.

From eqt. (4), with the given Θ , $\Phi_O^{-1}(y)$ is the linear equation $\sum_{j=1}^J w_j^O a_j = y - w_0^O$, which is a hyperplane in the activation space. As y changes, $\Phi_O^{-1}(y)$ forms parallel hyperplanes in the activation space; for any change of the same magnitude in y , the corresponding hyperplanes are spaced by the same distance. The activation space is entirely covered by these parallel $\Phi_O^{-1}(y)$ hyperplanes, orderly in terms of the values of y . These parallel hyperplanes form a (linear) scalar field [14], that is, for each point \mathbf{a} of the activation space, there is only one output value y whose $\Phi_O^{-1}(y)$ hyperplane passes point \mathbf{a} ; all points on the same (internal preimage) hyperplane yield the same y value.

From eqt. (5), $\Phi_A^{-1}(\mathbf{a})$ is a separable function such that each of its components lies along a dimension of the activation space. Moreover, $\Phi_{A_j}^{-1}(a_j) \equiv \{\mathbf{x} \in \Re^I | \sum_{i=1}^I w_{ji}^H x_i = \tanh^{-1}(a_j) - w_{j0}^H\}$ is a monotone bijection that defines a one-to-one mapping between the activation value a_j and the input \mathbf{x} . For each a_j value, $\Phi_{A_j}^{-1}(a_j)$ defines an activation hyperplane in the input space. Activation hyperplanes associated with all possible a_j values are parallel and form a (linear) scalar activation field in the input space. That is, for each point \mathbf{x} of the input space, there is only one activation value a_j whose $\Phi_{A_j}^{-1}(a_j)$ hyperplane passes point \mathbf{x} ; all points on the $\Phi_{A_j}^{-1}(a_j)$ hyperplane are associated with the activation value a_j . Each hidden node gives rise to an activation field, and J hidden nodes set up J independent activation fields in the input space. Thus, with a given Θ , the preimage of an activation value \mathbf{a} by Φ_A^{-1} is the intersection of J specific hyperplanes.

The intersection $\cap_{j=1}^J \{\mathbf{x} \in \Re^I | \sum_{i=1}^I w_{ji}^H x_i = \tanh^{-1}(a_j) - w_{j0}^H\}$ can be represented as $\{\mathbf{x} | \mathbf{W}^H \mathbf{x} = \boldsymbol{\omega}(\mathbf{a})\}$, where $\omega_j(a_j) \equiv \tanh^{-1}(a_j) - w_{j0}^H$ for all $1 \leq j \leq J$, and $\boldsymbol{\omega}(\mathbf{a}) \equiv (\omega_1(a_1), \omega_2(a_2), \dots, \omega_J(a_J))^T$. Given Θ and an arbitrary point \mathbf{a} , $\boldsymbol{\omega}(\mathbf{a})$ is simply a J -dimensional vector of known component values; the conditions that relates \mathbf{a} with \mathbf{x} can be represented as

$$\mathbf{W}^H \mathbf{x} = \boldsymbol{\omega}(\mathbf{a}), \quad (6)$$

which is a system of J simultaneous linear equations with I unknowns.

Let $\text{rank}(\mathbf{D})$ be the rank of matrix \mathbf{D} and $(\mathbf{D}_1 : \mathbf{D}_2)$ be the augmented matrix of two matrices \mathbf{D}_1 and \mathbf{D}_2 (with the same number of rows). $\mathbf{W}^H \mathbf{x} = \boldsymbol{\omega}(\mathbf{a})$ is a set of inconsistent simultaneous equations if $\text{rank}(\mathbf{W}^H : \boldsymbol{\omega}(\mathbf{a})) = \text{rank}(\mathbf{W}^H) + 1$ (c.f. [11]). In this case, the corresponding point \mathbf{a} is void. Otherwise, \mathbf{a} is non-void. Note that, for a non-void \mathbf{a} , the solution of eqt. (6) defines an affine space of dimension $I - \text{rank}(\mathbf{W}^H)$ in the input space. The discussion establishes Lemma 1 below.

Lemma 1. (a) An activation point \mathbf{a} in the activation space is non-void if its corresponding $\text{rank}(\mathbf{W}^H : \boldsymbol{\omega}(\mathbf{a}))$ equals $\text{rank}(\mathbf{W}^H)$. (b) The set of input values \mathbf{x} mapped onto a non-void \mathbf{a} forms an affine space of dimension $I - \text{rank}(\mathbf{W}^H)$ in the input space.

By definition, the non-void set equals $\{\mathbf{a} \in (-1, 1)^J | a_j = \tanh(\sum_{i=1}^I w_{ji}^H x_i + w_{j0}^H) \text{ for } 1 \leq j \leq J, \mathbf{x} \in \Re^I\}$. Check that \mathbf{W}^H is a $J \times I$ matrix. If $\text{rank}(\mathbf{W}^H) = J$, Lemma 1 says that no activation point \mathbf{a} can be void and leads to Lemma 2 below. For $\text{rank}(\mathbf{W}^H) < J$, Lemma 3 characterizes the non-void set, which requires the concept of manifold. A p -manifold is a Hausdorff space \mathbf{X} with a countable basis such that each point x of \mathbf{X} has a neighborhood that is homomorphic with an open subset of \Re^p [10]. A 1-manifold is often called a curve, and a 2-manifold is called a surface. For our purpose, it suffices to consider Euclidean spaces, the most common members of the family of Hausdorff spaces.

Lemma 2. If $\text{rank}(\mathbf{W}^H)$ equals J , then the non-void set covers the entire activation space.

Lemma 3. If $\text{rank}(\mathbf{W}^H)$ is less than J , then the non-void set in the activation space is a $\text{rank}(\mathbf{W}^H)$ -manifold.

$\mathbf{A}(y)$, the intersection of $\Phi_O^{-1}(y)$ and the non-void set in the activation space, is the internal-preimage of y . Mathematically, for each non-void y , $\mathbf{A}(y) \equiv \{\mathbf{a} | \text{rank}(\mathbf{W}^H \cdot \omega(\mathbf{a})) = \text{rank}(\mathbf{W}^H), \mathbf{a} \in \Phi_O^{-1}(y)\}$. Consider first $\text{rank}(\mathbf{W}^H) = J$. In this case, Lemma 2 says that the non-void set is the entire activation space. Thus, $\mathbf{A}(y)$ equals $\Phi_O^{-1}(y)$. If $\text{rank}(\mathbf{W}^H) < J$, then $\mathbf{A}(y)$ is a subset of $\Phi_O^{-1}(y)$. Thus, we have the following Lemma 4. Furthermore, $\mathbf{A}(y)$'s are aligned orderly according to $\Phi_O^{-1}(y)$ and all non-empty $\mathbf{A}(y)$'s form an *internal-preimage field* in the activation space. That is, there is one and only one y such that a non-void $\mathbf{a} \in \mathbf{A}(y)$; and for any \mathbf{a} on $\mathbf{A}(y)$, its output value is equal to y .

Lemma 4. For each non-void output value y , all points in the set $\mathbf{A}(y)$ are at the same hyperplane.

Now the preimage of any non-void output value y , $f^{-1}(y)$, equals $\{\mathbf{x} \in \mathbb{R}^I | \mathbf{W}^H \mathbf{x} = \omega(\mathbf{a}) \text{ with all } \mathbf{a} \in \mathbf{A}(y)\}$. If $\text{rank}(\mathbf{W}^H) = J$, then, from Lemma 2 and Lemma 1(b), the preimage $f^{-1}(y)$ is a $(I - 1)$ -manifold in the input space. For $\text{rank}(\mathbf{W}^H) < J$, from Lemma 3 and Lemma 1(b),

1. if $\text{rank}(\mathbf{W}^H) = 1$ and $\mathbf{A}(y)$ is a single point, then $f^{-1}(y)$ is a single hyperplane;
2. if $\text{rank}(\mathbf{W}^H) = 1$ and $\mathbf{A}(y)$ consists of several points, then $f^{-1}(y)$ may consist of several disjoint hyperplanes;
3. if $1 < \text{rank}(\mathbf{W}^H) < J$ and $\mathbf{A}(y)$ is a single $(\text{rank}(\mathbf{W}^H) - 1)$ -manifold, then $f^{-1}(y)$ is a single $(I - 1)$ -manifold; and
4. if $1 < \text{rank}(\mathbf{W}^H) < J$ and $\mathbf{A}(y)$ consists of several disjoint $(\text{rank}(\mathbf{W}^H) - 1)$ -manifolds, then $f^{-1}(y)$ consists of several disjoint $(I - 1)$ -manifolds.

Table 1 summarizes that the preimage $f^{-1}(y)$ is dictated by the property of its associated internal-preimage $\mathbf{A}(y)$.

Table 1. The relationship between the internal-preimage $\mathbf{A}(y)$ and the preimage $f^{-1}(y)$ of a non-void output value y

The nature of $\mathbf{A}(y)$	The nature of $f^{-1}(y)$
A single intersection-segment	A single $(I - 1)$ -manifold
Multiple disjoint intersection-segments	Multiple disjoint $(I - 1)$ -manifolds

The input space is entirely covered by a grouping of preimage manifolds that forms a *preimage field*. That is, there is one and only one preimage manifold passing through each \mathbf{x} ; and the corresponding output value is the y value associated with this preimage manifold. Note that the preimage manifolds are aligned orderly because $\mathbf{A}(y)$'s are aligned orderly according to $\Phi_O^{-1}(y)$'s and the mapping of Φ_A^{-1} is a monotone bijection that defines a one-to-one mapping between an activation vector and an affine space.

Notice that $\text{rank}(\mathbf{W}^H)$ determines the characteristic of the non-void set and thus the characteristic of internal-preimage. Hereafter, SLFN- p denotes a SLFN whose $\text{rank}(\mathbf{W}^H)$ equals p . For instance, if we adopt a SLFN-1 network, then we can assume \mathbf{w}_j^H equals $\alpha_j \mathbf{w}$ with $\alpha_j \neq 0$ for all j and $\alpha_{j_1} \neq \alpha_{j_2}$ for all $j_1 \neq j_2$. Since α_1 is non-zero, a_j can be represented as $\tanh(\delta_j \tanh^{-1}(a_1) + \delta_{j0})$, where $\delta_j = \alpha_j / \alpha_1$ and $\delta_{j0} = (\alpha_1 w_{j0}^H - \alpha_j w_{10}^H) / \alpha_1$. If we adopt a SLFN-2 network, then we can assume that \mathbf{w}_1^H equals \mathbf{w}_1 , \mathbf{w}_2^H equals \mathbf{w}_2 , $\mathbf{w}_j^H = \gamma_{j1} \mathbf{w}_1 + \gamma_{j2} \mathbf{w}_2$ with either γ_{j1} or γ_{j2} nonzero for all $j \geq 3$, and \mathbf{w}_1 and \mathbf{w}_2 are linearly independent.

For SLFN-1, the above preimage analysis states that the non-void set is an 1-manifold; $\mathbf{A}(y)$ equals $\{\mathbf{a} \in (-1, 1)^J | w_1^O a_1 + \sum_{j=2}^J w_j^O \tanh(\delta_j \tanh^{-1}(a_1) + \delta_{j0}) = y - w_0^O, a_j = \tanh(\delta_j \tanh^{-1}(a_1) + \delta_{j0}) \forall j \geq 2, a_1 \in (-1, 1)\}$; and $f^{-1}(y)$ equals $\{\mathbf{x} \in \Re^I | \sum_{i=1}^I w_{1i}^H x_i = \tanh^{-1}(a_1) - w_{10}^H, w_1^O a_1 + \sum_{j=2}^J w_j^O \tanh(\delta_j \tanh^{-1}(a_1) + \delta_{j0}) = y - w_0^O, a_j = \tanh(\delta_j \tanh^{-1}(a_1) + \delta_{j0}) \forall j \geq 2, a_1 \in (-1, 1)\}$. These establish the following Lemma 5. Furthermore, \mathbf{w} is the normal vector of the preimage hyperplane of SLFN-1 and $\mathbf{w}_j^H \equiv \alpha_j \mathbf{w}$ determines the orientation of the activation hyperplane in the input space corresponding to the j^{th} hidden node. Thus, we have Lemma 6.

Lemma 5. For SLFN-1, the preimage field is formed from a collection of preimage hyperplanes.

Lemma 6. For SLFN-1, the activation hyperplanes in the input space corresponding to all hidden nodes are parallel, and the preimage hyperplane is parallel with the activation hyperplane.

The above preimage analysis results in the following two hyperplane characteristics (i) and (ii) regarding all SLFN networks and one hyperplane characteristic (iii) for SLFN-1: (i) training samples with the same target value are allowed to be on the same activation hyperplane; (ii) activation points with the same target value are allowed to be on the same Φ_O^{-1} hyperplane; (iii) training samples with the same target value are allowed to be on the same preimage hyperplane.

4 The Experiment of m -Bit Parity Learning Problem

Through the application to the m -bit parity problem learned by constructing SLFN-2 and SLFN-1 network solutions, we show that the construction concept can help find network solutions perfectly fitting 2^m distinct samples with fewer used hidden nodes than the one obtained from the conventional construction method of [5].

For the m -bit parity problem, I is equal to m and we observe that the 2^m input samples are on the vertices of an m -dimensional hypercube with any two adjacent vertices having different target values. Without any loss of generality, take $x_i^c \in \{-1, 1\}$ for all c and i , and set the target value to t for odd number of +1's in input, and to $-t$ otherwise.

In the case of constructing a SLFN-2 network solution, assume that \mathbf{w}_1^H equals \mathbf{w}_1 , \mathbf{w}_2^H equals \mathbf{w}_2 , $\mathbf{w}_j^H \equiv \gamma_{j1} \mathbf{w}_1 + \gamma_{j2} \mathbf{w}_2$ with either γ_{j1} or γ_{j2} nonzero for

Table 2. The mapping from the total eight input patterns to the following six activation vectors

\mathbf{x}^c	$\mathbf{w}_1^T \mathbf{x}^c$	$\mathbf{w}_2^T \mathbf{x}^c$	$\mathbf{w}_j^T \mathbf{x}^c$	t^c
(1, 1, 1)	3	1	$3\gamma_{j1} + \gamma_{j2}$	-t
(-1, 1, 1)	1	3	$\gamma_{j1} + 3\gamma_{j2}$	t
(1, -1, 1)	1	-1	$\gamma_{j1} - \gamma_{j2}$	t
(1, 1, -1)	1	-1	$\gamma_{j1} - \gamma_{j2}$	t
(-1, -1, 1)	-1	1	$-\gamma_{j1} + \gamma_{j2}$	-t
(-1, 1, -1)	-1	1	$-\gamma_{j1} + \gamma_{j2}$	-t
(1, -1, -1)	-1	-3	$-\gamma_{j1} - 3\gamma_{j2}$	-t
(-1, -1, -1)	-3	-1	$-3\gamma_{j1} - \gamma_{j2}$	t

all $j \geq 3$, and \mathbf{w}_1 and \mathbf{w}_2 are linearly independent. Following the guide of the observed property of parity problem and the hyperplane characteristic (i), let \mathbf{w}_1 and \mathbf{w}_2 be assigned as in eqt. (6) and thus the total 2^m input patterns are mapped onto $2m$ activation vectors $\{\tilde{\mathbf{a}}^0, \dots, \tilde{\mathbf{a}}^{2m-1}\}$, in which $\tilde{\mathbf{a}}^k \equiv (\tilde{a}_1^k, \dots, \tilde{a}_J^k)^T$. Table 2 illustrates the mapping regarding the 3-bit parity problem. Let $\tilde{a}_1^0 = \tanh(m + w_{10}^H)$, $\tilde{a}_2^0 = \tanh(m - 2 + w_{20}^H)$, $\tilde{a}_j^0 = \tanh(m\gamma_{j1} + (m - 2)\gamma_{j2} + w_{j0}^H)$, $j = 3, \dots, J$, and $\tilde{t}^0 \equiv -t$; for each $k = 1, \dots, m - 1$, $\tilde{a}_1^{2k-1} = \tanh(m - 2k + w_{10}^H)$, $\tilde{a}_2^{2k-1} = \tanh(m - 2k + 2 + w_{20}^H)$, $\tilde{a}_j^{2k-1} = \tanh((m - 2k)\gamma_{j1} + (m - 2k + 2)\gamma_{j2} + w_{j0}^H)$, $j = 3, \dots, J$, and $\tilde{t}^{2k-1} \equiv (-1)^{k+1}t$; $\tilde{a}_1^{2k} = \tanh(m - 2k + w_{10}^H)$, $\tilde{a}_2^{2k} = \tanh(m - 2k - 2 + w_{20}^H)$, $\tilde{a}_j^{2k} = \tanh((m - 2k)\gamma_{j1} + (m - 2k - 2)\gamma_{j2} + w_{j0}^H)$, $j = 3, \dots, J$, and $\tilde{t}^{2k} \equiv (-1)^{k+1}t$; $\tilde{a}_1^{2m-1} = \tanh(-m + w_{10}^H)$, $\tilde{a}_2^{2m-1} = \tanh(-m + 2 + w_{20}^H)$, $\tilde{a}_j^{2m-1} = \tanh(-m\gamma_{j1} + (-m + 2)\gamma_{j2} + w_{j0}^H)$, $j = 3, \dots, J$, and $\tilde{t}^{2m-1} \equiv (-1)^{m+1}t$.

$$w_{1j} = 1, i = 1, \dots, m; \quad w_{21} = -1, w_{2i} = 1, i = 2, \dots, m. \quad (7)$$

Thus let $J = 2m$, $w_{j0}^H = 0 \forall j$, $w_0^O = 0$, and $\mathbf{w}^O = \widetilde{\mathbf{M}}^{-1} \widetilde{\mathbf{T}}$, in which $\widetilde{\mathbf{M}} \equiv (\tilde{\mathbf{a}}^0, \dots, \tilde{\mathbf{a}}^{2m-1})^T$ and $\widetilde{\mathbf{T}} \equiv (\tilde{t}^0, \tilde{t}^1, \dots, \tilde{t}^{2m-1})^T$. Referring to [5], it is trivial to show that there exist non-zero values of γ_{j1} and γ_{j2} such that the square matrix $\widetilde{\mathbf{M}}$ is invertible and thus the corresponding inverse matrix $\widetilde{\mathbf{M}}^{-1}$ exists. By checking all 2^m samples, it is trivial to show that the above SLFN-2 network is a solution of the m -bit parity problem.

In the case of constructing a SLFN-1 network solution, assume that \mathbf{w}_j^H equals $\alpha_j \mathbf{w}$ with $\alpha_j \neq 0$ for all j , $\alpha_{j_1} \neq \alpha_{j_2}$ for all $j_1 \neq j_2$, and a_j can be represented as $\tanh(\delta_j \tanh^{-1}(a_1) + \delta_{j0})$, where $\delta_j = \alpha_j / \alpha_1$ and $\delta_{j0} = (\alpha_1 w_{j0}^H - \alpha_j w_{10}^H) / \alpha_1$. Following the guide of the observed property of parity problem and the hyperplane characteristics (ii) and (iii), we pick \mathbf{w} as \mathbf{w}_1 in eqt. (6) and assign a total of $\lceil (m+1)/2 \rceil$ adopted hidden nodes. Thus the total 2^m input patterns are mapped onto $m + 1$ activation vectors, $\{\hat{\mathbf{a}}^0, \dots, \hat{\mathbf{a}}^m\}$, in which $\hat{a}_j^k = \tanh((m - 2k)\alpha_j + w_{j0}^H)$, $j = 1, \dots, \lceil (m+1)/2 \rceil$ and $\hat{\mathbf{a}}^k \equiv (\hat{a}_1^k, \dots, \hat{a}_{\lceil (m+1)/2 \rceil}^k)^T$, $k = 0, \dots, m$. Then we make the following assignments:

- (I) when m is an odd number: let $w_{j0}^H = 0 \forall j, w_0^O = 0$ and $\mathbf{w}^O = \widehat{\mathbf{M}}^{-1}\widehat{\mathbf{T}}$, where $\widehat{\mathbf{M}} \equiv (\widehat{\mathbf{a}}^0, \widehat{\mathbf{a}}^1, \dots, \widehat{\mathbf{a}}^{\lceil(m+1)/2\rceil-1})^T$, $\widehat{a}_j^k = \tanh((m-2k)\alpha_j) \forall j, k = 0, \dots, m$, $\widehat{\mathbf{T}} \equiv (\widehat{t}^0, \widehat{t}^1, \dots, \widehat{t}^{\lceil(m+1)/2\rceil-1})^T$, and $\widehat{t}^k \equiv (-1)^{k+1}t$ for all k .
- (II) when m is an even number: let $w_{j0}^H = \alpha_j \forall j, w_0^O = 0$ and $\mathbf{w}^O = \widehat{\mathbf{M}}^{-1}\widehat{\mathbf{T}}$, where $\widehat{\mathbf{M}} \equiv (\widehat{\mathbf{a}}^0, \widehat{\mathbf{a}}^1, \dots, \widehat{\mathbf{a}}^{\lceil(m+1)/2\rceil-1})^T$, $\widehat{a}_j^k = \tanh((m-2k+1)\alpha_j) \forall j, k = 0, \dots, m$, $\widehat{\mathbf{T}} \equiv (\widehat{t}^0, \widehat{t}^1, \dots, \widehat{t}^{\lceil(m+1)/2\rceil-1})^T$, and $\widehat{t}^k \equiv (-1)^{k+1}t$ for all k .

Referring to [5], it is trivial to show that there exist non-zero values of α_j such that the set $\{\widehat{\mathbf{a}}^0, \dots, \widehat{\mathbf{a}}^m\}$ are linearly independent and thus the square matrix $\widehat{\mathbf{M}}$ is invertible. By checking all 2^m samples, it is trivial to show that the above SLFN-1 network is a solution of the m -bit parity problem.

5 Conclusions and Future Work

This study derives three hyperplane characteristics and several properties of the SLFN through the preimage analysis. Regarding the m -bit parity learning problem, we observe that the 2^m input samples are on the vertices of an m -dimensional hypercube with any two adjacent vertices having different target values. Accordingly, the construction concept helps set up SLFN-1 and SLFN-2 solutions, each of which uses fewer hidden nodes than the ones used by the conventional construction method of [5].

Note that most learning algorithms (or construction methods) lead to a SLFN- p solution with $p \geq 2$ and complex preimages. Extending from this study, one may further argue that the construction concept can help identify the *true* upper bound on the number of used hidden nodes of such SLFN solutions for a specific learning problem. This argument is one of future researches. Most training is a trade-off of learning performance and generalization performance, and the hidden layer plays a key role in this issue. Therefore, another future research is to develop and validate a construction concept that involves with the generalization.

Acknowledgments. This study is supported by the National Science Council of the R.O.C. under Grants NSC 92-2416-H-004-004, NSC 93-2416-H-004-015, and NSC 43028F.

References

1. Arai, M.: Bounds on the number of hidden units in binary-valued three-layer neural networks. *Neural Networks* 6, 855–860 (1993)
2. Arslanov, M.Z., Ashigaliev, D.U., Ismail, E.E.: N-bit parity ordered neural networks. *Neurocomputing* 48, 1053–1056 (2002)
3. Hertz, J., Krogh, A., Palmer, R.: *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Redwood City (1991)

4. Hohil, M.E., Liu, D.R., Smith, S.H.: Solving the N-bit parity problem using neural networks. *Neural Networks* 12(11), 1321–1323 (1999)
5. Huang, G., Babri, H.: Upper bounds on the number of hidden neurons in feed-forward networks with arbitrary bounded nonlinear activation functions. *IEEE Transactions on Neural Networks* 9, 224–229 (1998)
6. Huang, S.C., Huang, Y.F.: Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Transactions on Neural Networks* 2, 47–55 (1991)
7. Iyoda, E.M., Nobuhara, H., Hirota, K.: A solution for the N-bit parity problem using a single translated multiplicative neuron. *Neural Processing Letters* 18(3), 213–218 (2003)
8. Lavretsky, E.: On the exact solution of the Parity-N problem using ordered neural networks. *Neural Networks* 13(8), 643–649 (2000)
9. Liu, D.R., Hohil, M.E., Smith, S.H.: N-bit parity neural networks: new solutions based on linear programming. *Neurocomputing* 48, 477–488 (2002)
10. Munkres, J.: Topology: a first course. Prentice-Hall, Englewood Cliffs (1975)
11. Murty, K.: Linear Programming. John Wiley & Sons, NY (1983)
12. Setiono, R.: On the solution of the parity problem by a single hidden layer feed-forward neural network. *Neurocomputing* 16, 225–235 (1997)
13. Sontag, E.: Feedforward nets for interpolation and classification. *J. Comput. System Sci.* 45, 20–48 (1992)
14. Tsaih, R.: An explanation of reasoning neural networks. *Mathematical and Computer Modelling* 28, 37–44 (1998)
15. Urcid, G., Ritter, G.X., Iancu, L.: Single layer morphological Perceptron solution to the N-bit parity problem. In: Sanfeliu, A., Martínez Trinidad, J.F., Carrasco Ochoa, J.A. (eds.) CIARP 2004. LNCS, vol. 3287, pp. 171–178. Springer, Heidelberg (2004)

Comparative Study of the CG and HBF ODEs Used in the Global Minimization of Nonconvex Functions

Amit Bhaya, Fernando A. Pazos, and Eugenius Kaszkurewicz

Department of Electrical Engineering - COPPE/UFRJ

C.P. 68504, 21945/970 – Rio de Janeiro, RJ, Brazil

amit@nacad.ufrj.br, quini@ort.org.br, eugenius@nacad.ufrj.br

Abstract. This paper presents a unified *control Liapunov function* (CLF) approach to the design of heavy ball with friction (HBF) and conjugate gradient (CG) neural networks that aim to minimize scalar nonconvex functions that have continuous first- and second-order derivatives and a unique global minimum. This approach leads naturally to the design of second-order differential equations which are the mathematical models of the corresponding implementations as neural networks. Preliminary numerical simulations indicate that, on a small suite of benchmark test problems, a continuous version of the well known conjugate gradient algorithm, designed by the proposed CLF method, has better performance than its HBF competitor.

Keywords: Control Liapunov Functions, Nonconvex Functions, Global Minimization, Second Order ODEs.

1 Introduction

Neural networks for optimization are typically implementations of dynamical systems represented by ordinary differential equations (ODE) and this paper will use the terms ODE and neural network synonymously. There has been a revival of interest in analog or ordinary differential equation based methods for optimization, dating from the seminal paper of Karmarkar and the subsequent boom in interior point and trajectory-following methods (see, for example, [1][2][3] and the references therein).

Methods to solve different optimization problems arise in a unified and natural manner by taking a control viewpoint. This control perspective was presented in [3][4][5]. In the case of minimizing a scalar function, it consists of formulating the problem as that of designing a closed-loop control system, with a plant defined by the scalar function and a feedback controller: the overall system is modeled by an ODE whose trajectories converge to the desired minimum of the function. The choice of the feedback controller is based on the control Liapunov function (CLF) method (detailed in the present context in the book [3]), well known in control theory. This control perspective offers powerful tools for the design and analysis of systems to solve optimization problems.

The main motivation to use second-order ODEs (where second-order refers to the fact that the overall or closed-loop dynamics is described by a second-order ODE), instead of first-order ones, comes from the fact that, for nonconvex functions $\phi(\mathbf{x})$:

$\mathbb{R}^n \rightarrow \mathbb{R}$, trajectories $\mathbf{x}(t)$ starting at an initial point $\mathbf{x}(0) = \mathbf{x}_0 \in \text{dom}(\phi)$ and generated by first-order gradient-based ODEs will converge to any point where the gradient of the function is zero, even though this point may be a local minimum of the function; using second-order ODEs, with an adequate choice of parameters (gains), trajectories are able to converge to the global minimum of the function \mathbf{x}^* . These ODEs are suitable for implementation as neural networks.

In [6] several second order ODE's were interpreted and designed as closed-loop control system using CLFs. The ODE's studied in [6] were the *heavy ball with friction* (HBF) method, proposed by Polyak [7] and later studied in detail in [8][9][10], the *dynamical inertial Newton-like system* (DIN), proposed by Alvarez et. al. [11], and a continuous-time version of the *conjugate gradient* (CG) method. In addition, other second order systems to minimize nonconvex scalar functions were also designed using the CLF method, by choosing of different candidate Liapunov functions and with adequate choices of the plant and the controller of the systems.

In [6], it was noted that the continuous time CG algorithm was superior to most of the others with the chosen test functions. In this respect, the contribution of this paper is to go beyond, showing that the much studied HBF method and some of its variants [8][9][10], as well as generalizations proposed in this paper are outperformed on a suite of standard nonconvex problems used in the literature on global optimization by a continuous version of the CG algorithm that was first proposed for quadratic functions in [12], and generalized in [6].

In order to proceed, consider a scalar function $\phi(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$, continuous and with continuous partial derivatives, and radially unbounded (that is, $\phi(\mathbf{x}) \rightarrow \infty$ when $\|\mathbf{x}\| \rightarrow \infty$), such that for all $\mathbf{x} \in \text{dom}(\phi)$, $\phi(\mathbf{x}) \geq \phi(\mathbf{x}^*) = \phi_{\min}$, i.e. the global minimum \mathbf{x}^* is assumed to exist and be unique. It is desired to find \mathbf{x}^* , by following an ODE trajectory that starts from some given initial point \mathbf{x}_0 .

2 Preliminaries: Motivation and CLF-Based Design

In order to arrive at a natural motivation for the approach proposed in this paper, first consider a continuous version of the steepest descent (SD) algorithm, written as:

$$\dot{\mathbf{x}}(t) = \mathbf{u} \quad (1)$$

$$\mathbf{u} = -\nabla\phi(\mathbf{x}(t)) , \quad (2)$$

where $\mathbf{u} \in \mathbb{R}^n$ and the gradient of ϕ is written $\nabla\phi : \mathbb{R}^n \rightarrow \mathbb{R}^n : \mathbf{x} \mapsto \nabla\phi(\mathbf{x})$. The reason that (1) and (2) are written in this fashion is twofold: first, it is clear that its trajectories will converge to local minima satisfying the necessary condition $\nabla\phi(\mathbf{x}(t)) = 0$ and second, in order to escape from confinement to a local minimum, a more sophisticated choice of \mathbf{u} (which is being thought of as the control) is needed in order to drive \mathbf{x} to the global minimum \mathbf{x}^* ($\phi(\cdot)$ is being thought of as defining the output of the controlled object or plant).

Assume that \mathbf{u} has first-order dynamics, i.e., (2) is replaced by:

$$\dot{\mathbf{u}} = \psi(\mathbf{x}) . \quad (3)$$

Then, a natural CLF arises from the attempt to solve the so called regulation problem, in which the reference signal is the global minimum value of the function ϕ_{\min} , and the output of the plant, $\phi(\mathbf{x})$, is to be driven to this value. This is to be done by choice of the RHS $\psi(\mathbf{x})$ of the \mathbf{u} -dynamics (3), also ensuring that \mathbf{u} goes to zero, so that the \mathbf{x} -dynamics (1) converges to the desired global equilibrium. In other words, a candidate CLF can be written as follows.

$$V(\mathbf{x}, \mathbf{u}) = \phi(\mathbf{x}) - \phi_{\min} + \frac{\mathbf{u}^T \mathbf{u}}{2} \geq 0 , \quad (4)$$

which is greater than zero for all $\mathbf{x} \neq \mathbf{x}^*$ and $\mathbf{u} \neq \mathbf{0}$.

The time derivative of (4) is:

$$\dot{V}(\mathbf{x}, \mathbf{u}) = \nabla \phi(\mathbf{x})^T \dot{\mathbf{x}} + \mathbf{u}^T \dot{\mathbf{u}} . \quad (5)$$

Equations (1), (3), (4) and (5) are the starting point for the CLF design of new second-order ODEs by choosing $\dot{\mathbf{x}}$ and $\dot{\mathbf{u}}$ in (5) (respectively, the plant dynamics and the controller dynamics in control jargon) so that $\dot{V}(\mathbf{x}, \mathbf{u}) \leq 0$, ensuring stability. The name CLF is justified by the fact that \dot{V} is made negative by choice of the control or \mathbf{u} -dynamics, given some choice of the plant or \mathbf{x} -dynamics.

An important observation is that, although the global minimum value of the function (ϕ_{\min}) occurs in the definition of the candidate CLF (4), it disappears in its derivative (5), which is the crucial equation for the design. In other words, knowledge of ϕ_{\min} is not required for the CLF design.

3 CLF Design of the CG and HBF ODEs

3.1 The HBF ODE

The best known existing algorithm of the class of so called second-order algorithms is the “heavy ball with friction” (HBF), represented by the following second-order differential equation:

$$\ddot{\mathbf{x}} + \gamma \dot{\mathbf{x}} + \nabla \phi(\mathbf{x}) = \mathbf{0} , \quad (6)$$

where γ is positive scalar coefficient. In a mechanical analogy of (6), the HBF ODE represents a dissipative system, the scalar γ being interpreted as viscous friction. A natural generalization of (6) consists of introduction of scalar coefficients that depend on \mathbf{x} and $\dot{\mathbf{x}}$, and this nonlinear system will be denoted as gHBF:

$$\ddot{\mathbf{x}} + \gamma(\mathbf{x}, \dot{\mathbf{x}}) \dot{\mathbf{x}} + \mu(\mathbf{x}, \dot{\mathbf{x}}) \nabla \phi(\mathbf{x}) = \mathbf{0} , \quad (7)$$

where $\gamma(\mathbf{x}, \dot{\mathbf{x}})$, $\mu(\mathbf{x}, \dot{\mathbf{x}}) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ are scalar valued functions of the state vector $[\mathbf{x}^T \dot{\mathbf{x}}^T]^T$.

Equation (7) has the following standard realization as a first order vector ODE:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{u} \\ \dot{\mathbf{u}} &= -\gamma(\mathbf{x}, \mathbf{u}) \mathbf{u} - \mu(\mathbf{x}, \mathbf{u}) \nabla \phi(\mathbf{x}) . \end{aligned} \quad (8)$$

Note that the equilibrium point of (8) is $[\nabla^T \phi(\mathbf{x}) \mathbf{u}^T]^T = \mathbf{0}$, which implies that trajectories generated by this system can stop at any zero gradient point; however, the hope is that, with an adequate choice of the parameters, trajectories be able to pass local minima to converge to the global minimum.

Substituting (8) in (5) yields:

$$\dot{V} = (1 - \mu(\mathbf{x}, \mathbf{u}))\nabla^T \phi(\mathbf{x})\mathbf{u} - \gamma(\mathbf{x}, \mathbf{u})\mathbf{u}^T\mathbf{u}, \quad (9)$$

and it remains to choose $\mu(\mathbf{x}, \mathbf{u}), \gamma(\mathbf{x}, \mathbf{u})$ in such a way that \dot{V} in (9) becomes negative, at least semidefinite. If \dot{V} is uniformly continuous, but only semidefinite, other tools like La Salle's theorem or Barbalat's lemma can be used to prove convergence; otherwise, if \dot{V} is discontinuous but negative definite, analysis of convergence can be carried out using the tools of nonsmooth analysis [13]. The details are omitted here for lack of space and will be published elsewhere.

Two of the possible choices that make \dot{V} negative are presented below.

For example, suppose that γ is chosen as a fixed constant. Then, in order to make \dot{V} in (9) negative, $\mu(\mathbf{x}, \mathbf{u})$ can be switched as follows:

$$\mu(\mathbf{x}, \mathbf{u}) = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi(\mathbf{x}) \mathbf{u}} + a & \text{if } \nabla^T \phi(\mathbf{x}) \mathbf{u} > 0 \\ 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi(\mathbf{x}) \mathbf{u}} - b & \text{if } \nabla^T \phi(\mathbf{x}) \mathbf{u} < 0 \\ 1 & \text{if } \nabla^T \phi(\mathbf{x}) \mathbf{u} = 0, \end{cases} \quad (10)$$

where a and b are positive scalar values (not necessarily constants). For example, the choice $a = -b = \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi(\mathbf{x}) \mathbf{u}}$ leads to $\mu = 1$, which makes gHBF (7) equal to the net HBF (6).

If a and b are chosen as constants, the result of substituting (10) in (9) is

$$\begin{aligned} \text{if } \nabla^T \phi(\mathbf{x}) \mathbf{u} > 0 & \quad \dot{V} = -a \nabla^T \phi(\mathbf{x}) \mathbf{u} < 0 \\ \text{if } \nabla^T \phi(\mathbf{x}) \mathbf{u} < 0 & \quad \dot{V} = b \nabla^T \phi(\mathbf{x}) \mathbf{u} < 0 \\ \text{if } \nabla^T \phi(\mathbf{x}) \mathbf{u} = 0 & \quad \dot{V} \in \text{co}\{-a \nabla^T \phi(\mathbf{x}) \mathbf{u}, b \nabla^T \phi(\mathbf{x}) \mathbf{u}\} = 0, \end{aligned}$$

where co denotes the convex hull between scalar values. This implies that the equilibrium may be any point where $\nabla^T \phi(\mathbf{x}) \mathbf{u} = 0$, but from (8) and (4) it follows that the unique equilibrium point is $[\nabla^T \phi(\mathbf{x}) \mathbf{u}^T]^T = \mathbf{0}$.

The system (8) with γ constant and μ chosen as in (10) will be denoted as HBF1(γ, a, b).

Another possibility is to choose μ as a fixed constant, and in this case the choice $\gamma(\mathbf{x}, \mathbf{u}) > (1 - \mu) \frac{\nabla^T \phi(\mathbf{x}) \mathbf{u}}{\mathbf{u}^T \mathbf{u}}$ makes \dot{V} negative in (9).

To validate this inequality, it is enough to choose:

$$\gamma(\mathbf{x}, \mathbf{u}) = (1 - \mu) \frac{\nabla^T \phi(\mathbf{x}) \mathbf{u}}{\mathbf{u}^T \mathbf{u}} + a, \quad (11)$$

where a is a positive scalar. For example, the choice $\mu = 1$ also leads to the net HBF (6), with a parameter $\gamma = a$. This second choice (11), with the parameter μ constant, will be denoted as HBF2(μ, a).

3.2 The CG ODE

The conjugate gradient (CG) method is a well known discrete algorithm applied to unconstrained optimization. It is commonly used to minimize scalar convex quadratic functions. In [12], the authors show that the *back propagation with momentum* (BPM) method, much analyzed in the neural network literature, is a version of the conjugate gradient method for quadratic functions.

A natural approach to continuous time version of the CG method was first presented for quadratic functions in [12, p. 69], generalized in [6], and is restated below for convenience:

$$\begin{aligned}\dot{\mathbf{x}} &= \alpha(\mathbf{x}, \mathbf{u})\mathbf{u} \\ \dot{\mathbf{u}} &= -\nabla\phi(\mathbf{x}) - \beta(\mathbf{x}, \mathbf{u})\mathbf{u}.\end{aligned}\quad (12)$$

Equation (12) defines the *CG ODE*, so named to recall its origins as a continuous version of the conjugate gradient algorithm. It should be emphasized that, in the continuous time case (12), the letters CG are just a convenient mnemonic to recall the origin of the equation and not intended to draw attention to any conjugacy concepts.

Substituting (12) in (5) yields

$$\dot{V} = (\alpha(\mathbf{x}, \mathbf{u}) - 1)\mathbf{u}^T\nabla\phi(\mathbf{x}) - \beta(\mathbf{x}, \mathbf{u})\mathbf{u}^T\mathbf{u}. \quad (13)$$

Here too, there are many ways to make \dot{V} negative, and we mention just two.

In the first case we consider β as a fixed constant. To make \dot{V} negative in (13), the function $\alpha(\mathbf{x}, \mathbf{u})$ is chosen as:

$$\alpha(\mathbf{x}, \mathbf{u}) = \begin{cases} 1 + \beta \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} - a & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ 1 + \beta \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + b & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ 1 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0,\end{cases} \quad (14)$$

where a and b are positive scalars. Again, the choice $a = -b = \beta \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}}$ yields $\alpha = 1$, which makes the CG ODE (12) equal to the HBF ODE (6). A convergence analysis similar to that made in the former section proves convergence to the equilibrium point $[\nabla^T\phi(\mathbf{x}) \ \mathbf{u}^T]^T = \mathbf{0}$.

The ODE (12) with β constant and $\alpha(\mathbf{x}, \mathbf{u})$ chosen as (14) will be denoted as CG1(β, a, b).

There exists another possibility which consists of choosing of the parameter α as a constant different from zero. In this case, to make \dot{V} negative in (13), $\beta(\mathbf{x}, \mathbf{u}) > (\alpha - 1) \frac{\nabla^T\phi(\mathbf{x})\mathbf{u}}{\mathbf{u}^T\mathbf{u}}$. However, note that, in this case, (12) can be written as a second order ODE in the form $\ddot{\mathbf{x}} + \beta(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \alpha\nabla\phi(\mathbf{x}) = \mathbf{0}$, which becomes the net HBF2(μ, a), under the change of variables $\mathbf{r} = \alpha\mathbf{u}$.

Remark 1. Observe that, if $\alpha(\mathbf{x}, \mathbf{u})$ is assumed differentiable in \mathbf{x}, \mathbf{u} , then (12) can be rewritten (in terms of $\ddot{\mathbf{x}}, \dot{\mathbf{x}}, \mathbf{x}$) as follows:

$$\ddot{\mathbf{x}} + \beta(\mathbf{x}, \mathbf{u})\dot{\mathbf{x}} + \alpha(\mathbf{x}, \mathbf{u})\nabla\phi(\mathbf{x}) - \left[\frac{\partial\alpha}{\partial\mathbf{x}}\dot{\mathbf{x}} + \frac{\partial\alpha}{\partial\mathbf{u}}\dot{\mathbf{u}} \right] \mathbf{u} = \mathbf{0}.$$

If it is assumed that α depends only on \mathbf{x} (and not on \mathbf{u}), then (12) can be reduced to second order form, yielding (assuming $\alpha(\mathbf{x}) \neq 0$, for all \mathbf{x}):

$$\ddot{\mathbf{x}} + \left[\beta(\mathbf{x}) - \frac{\nabla \alpha(\mathbf{x})^T \dot{\mathbf{x}}}{\alpha(\mathbf{x})} \right] \dot{\mathbf{x}} + \alpha(\mathbf{x}) \nabla \phi(\mathbf{x}) = 0 , \quad (15)$$

so that the state-dependent term $\alpha(\mathbf{x})$ is seen to give rise to a nonlinear damping term (the term within square brackets in (15)). Of course, the switching law (14) proposed above, for the CG ODE, is discontinuous, so reduction to the second order form above is not possible, but it does serve as an indication that the CG dynamics are quite different and more complex than the HBF dynamics.

4 Computational Experiments

In this section, the algorithms formulated in the former section will be tested with different test functions commonly used in the literature. The algorithms will be implemented with SIMULINK of MATLAB 6, discretizing them by the forward Euler method with a fixed step size of $0.01s$ during $10s$.

4.1 Simulations with the Inverted Camelback Function

The inverted camelback function, popular in the literature on global optimization, was chosen as objective function:

$$\phi(x_1, x_2) = ax_1^2 + bx_1^4 + cx_1^6 - x_1x_2 + dx_2^2 + ex_2^4 , \quad (16)$$

with the following constants $a = -2$, $b = 1.05$, $c = -\frac{1}{6}$, $d = -1$ and $e = 0$. With these constants, the inverted camelback function has a global minimum at $\mathbf{x} = [0 \ 0]^T$, two local minima at $\mathbf{x} = [-1.7475 \ 0.8737]^T$ and $\mathbf{x} = [1.7475 \ -0.8737]^T$, and two saddle points at $\mathbf{x} = [-1.0705 \ 0.5352]^T$ and $\mathbf{x} = [1.0705 \ -0.5352]^T$.

Figure 1 shows the trajectories and the error norms generated by the different ODEs from the initial point $\mathbf{x}_0 = [2 \ 1]^T$. The initial value of the control variable was chosen as $\mathbf{u}_0 = [0 \ 0]^T$. The parameters γ for the HBF and HBF1 ODEs and β for the CG1 ODE were chosen as $\gamma = \beta = 1.7$. This is the maximum value for the trajectory generated by the HBF ODE to converge to the global minimum from this initial point and with this initial value of the control variable. The other parameters were chosen by hand tuning in such a way as to try to achieve the fastest convergence to the global minimum. These parameters are shown in Table 1. The trajectory generated by the CG1 ODE converges faster than the trajectories generated by its competitors.

Figure 2 shows the trajectories and the error norms generated by the different ODEs from the initial point $\mathbf{x}_0 = [-1.5 \ -1.5]^T$. The initial value of the control variable was chosen as $\mathbf{u}_0 = [1 \ 1]^T$ and the parameters used were $\gamma = \beta = 3$. Once again, the other parameters were chosen to generate the fastest convergence to the global minimum and are presented in Table 1.

Figure 2 shows that the trajectory generated by CG1 converges almost an order of magnitude faster than those generated by its competitors.

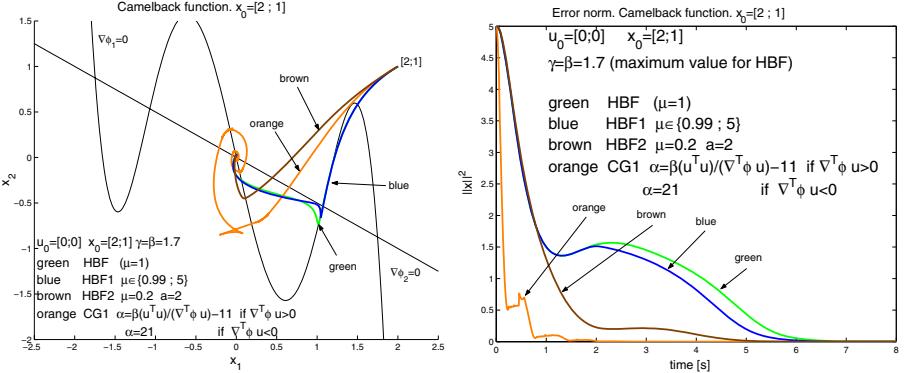


Fig. 1. Trajectories generated with the different algorithms and error norms with the camelback function from the initial point $x_0 = [2; 1]^T$. The initial value of the control variable was chosen as $u_0 = [0; 0]^T$ and the parameters chosen as $\gamma = \beta = 1.7$.

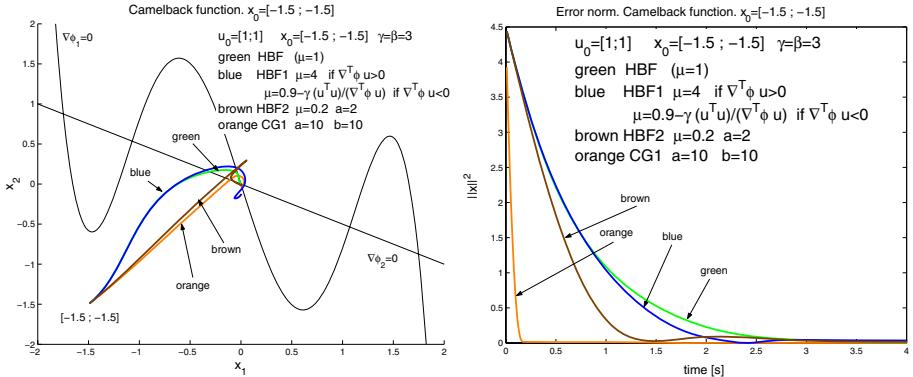


Fig. 2. Trajectories generated with the different algorithms and error norms with the camelback function from the initial point $x_0 = [-1.5; -1.5]^T$. The initial value of the control variable was chosen as $u_0 = [1; 1]^T$ and the parameters were chosen as $\gamma = \beta = 3$.

4.2 Simulations with the Rastrigin Function

The Rastrigin function was chosen as objective function:

$$\phi(\mathbf{x}) = \sum_{i=1}^N x_i^2 - \cos(2\pi x_i) : \mathbb{R}^N \rightarrow \mathbb{R}, \quad (17)$$

This function has several local minima and a unique global minimum at $\mathbf{x}^* = \mathbf{0}$. The value of the function at the minimum is $\phi(\mathbf{x}^*) = -N$, where N is the number of variables, and was chosen as 10 for the numerical experiments reported here. The initial point was chosen as $x_{0,i} = -1$ and the initial value of the control variable as $u_{0,i} = 6$ for all $i \in \{1, \dots, N\}$.

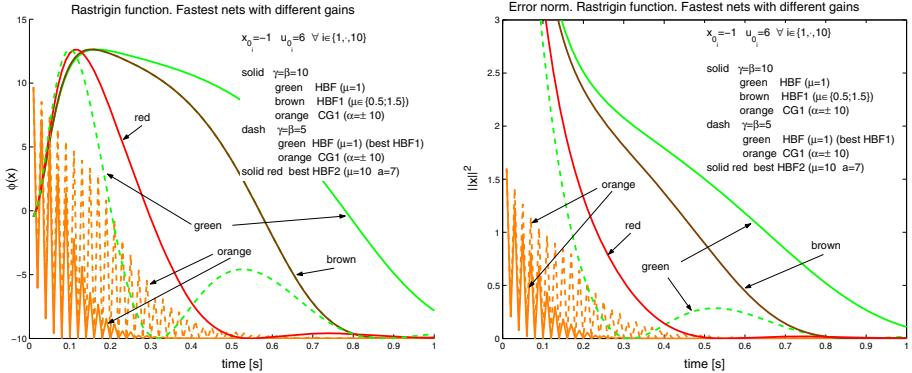


Fig. 3. Value of the function and error norms vs. time with the Rastrigin function for the trajectories generated by the different algorithms. The values of the parameters γ for HBF and HBF1, and β for CG1 were chosen as $\gamma = \beta = 10$ (solid curves) and $\gamma = \beta = 5$ (dashed curves). With $\gamma = 5$, the best value of μ for the HBF1 net was $\mu = 1$, which coincides with HBF; thus, dashed green and dashed brown curves coincide.

The ODEs were tested with different parameter values, but Fig. 3 shows only $\gamma = \beta = 10$ and $\gamma = \beta = 5$. The other parameters were chosen in such a way to generate the fastest convergence to the global minimum and are presented in Table 1.

Figure 3 shows that, with both of the selected parameters, the trajectory generated by the CG1 net converges an order of magnitude faster than those generated by the other ODEs.

4.3 Simulations with the Schwefel Function

The Schwefel function was chosen as objective function:

$$\phi(\mathbf{x}) = - \sum_{i=1}^N x_i \sin(\sqrt{|x_i|}) : \mathbb{R}^N \rightarrow \mathbb{R}, \quad (18)$$

where N is the number of variables, chosen as 10 for all numerical experiments reported here.

This function has several local minima and, inside the interval $x_i \in [-500, 500]$ it has an unique global minimum at $x_i^* = 420.9687$, for all $i \in \{1, \dots, N\}$. The value of the function at the minimum is $\phi(\mathbf{x}^*) = -4189.8$. The initial point was chosen as $x_{0,i} = 100$ and the initial value of the control variable as $u_{0,i} = 200$ for all $i \in \{1, \dots, N\}$.

The ODEs were tested with different values of γ and β , but Fig. 4 shows only $\gamma = \beta = 0.7$ and $\gamma = \beta = 0.6$. The other parameters were chosen by hand tuning to generate the fastest convergence to the global minimum and are presented in Table 1.

Figure 4 shows that, with both of the selected parameters, the trajectory generated by the CG1 net converges faster than those generated by the other ODEs.

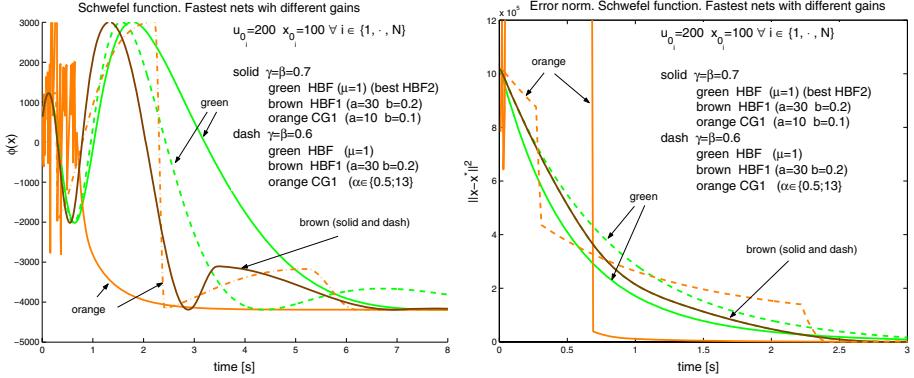


Fig. 4. Value of the function and error norms vs. time with the Schwefel function for the trajectories generated by the different algorithms. The parameters γ for HBF and HBF1, and β for CG1 were chosen as $\gamma = \beta = 0.7$ (solid curves) and $\gamma = \beta = 0.6$ (dashed curves). Trajectories generated by HBF1 are the same for both values of γ , thus, the solid brown curve coincides with the dashed brown one. The fastest trajectory generated by the HBF2 ODE coincides with that generated by the HBF ODE with $\gamma = 0.7$.

Table 1. Parameters used to achieve the fastest convergence to the global minimum

	Camelback		Rastrigin		Schwefel	
	$x_0 = [2 \ 1]^T$	$x_0 = [-1.5 \ -1.5]^T$	$x_{0i} = -1$	$u_{0i} = 6$	$x_{0i} = 100$	$u_{0i} = 200$
	$u_0 = [0 \ 0]^T$	$u_0 = [1 \ 1]^T$				
HBF	$\gamma = 1.7$	$\gamma = 3$	$\gamma = 10$	$\gamma = 5$	$\gamma = 0.7$	$\gamma = 0.6$
HBF1	$\gamma = 1.7$ $\mu \in \{0.99; 5\}$	$\gamma = 3$ $\mu \in \{4; 0.9 - \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}}\}$	$\gamma = 10$ $\mu \in \{0.5; 1.5\}$	$\gamma = 5$ $\mu = 1$	$\gamma = 0.7$ $\mu \in \{31 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}}; 0.8 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}}\}$	$\gamma = 0.6$ $\mu \in \{31 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}}; 0.8 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}}\}$
HBF2	$\mu = 0.2$ $\gamma = (1 - \mu) \frac{\nabla T \cdot \phi \mathbf{u}}{\mathbf{u}^T \mathbf{u}} + 2$	$\mu = 0.2$ $\gamma = (1 - \mu) \frac{\nabla T \cdot \phi \mathbf{u}}{\mathbf{u}^T \mathbf{u}} + 2$	$\mu = 10$ $\gamma = (1 - \mu) \frac{\nabla T \cdot \phi \mathbf{u}}{\mathbf{u}^T \mathbf{u}} + 7$		$\mu = 1$ $\gamma = 0.7$	
CG1	$\beta = 1.7$ $\alpha \in \{-11; 21\}$ $\{ \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}} - 11; 21 \}$	$\beta = 3$ $\alpha \in \{ \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}} - 9; \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}} + 11 \}$	$\beta = 10$ $\alpha \in \{-10; 10\}$	$\beta = 5$ $\alpha \in \{-10; 10\}$	$\beta = 0.7$ $\alpha \in \{ \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}} - 9; \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla T \cdot \phi \mathbf{u}} + 1.1 \}$	$\beta = 0.6$ $\alpha \in \{0.5; 13\}$

5 Conclusions

Of the different ODEs tested (HBF1,2, and CG1), the CG1(β, a, b) ODE (12) presented the best behavior on the suite of benchmark examples on which all algorithms were tested, showing fast convergence and good transients and accuracy. The CG1(β, a, b) method, with the \mathbf{x} -dynamics being directly affected by the switching law in $\alpha(\mathbf{x}, \mathbf{u})$ does better than the best HBF methods, because, in the former class of methods, the CLF designed switching law directly affects the \mathbf{u} -dynamics, and, as a consequence, the \mathbf{x} -dynamics.

In the light of the results obtained in this paper, it can be affirmed that the CG net proposed here performs better than the well known HBF net and its variants, which

have received much attention in the literature. Further research on initialization of the CG net so as to converge to the global minimum with a high probability needs to be carried out.

Finally, we emphasize that all the ODEs presented here can be easily implemented as neural networks.

References

1. Cichocki, A., Unbehauen, R.: Neural Networks for Optimization and Signal Processing. John Wiley, Chichester (1993)
2. Chu, M.T.: Linear algebra algorithms as dynamical systems. *Acta Numerica*, 1–86 (2008)
3. Bhaya, A., Kaszkurewicz, E.: Control perspectives on numerical algorithms and matrix problems. In: *Advances in Control*. SIAM, Philadelphia (2006)
4. Bhaya, A., Kaszkurewicz, E.: Iterative methods as dynamical systems with feedback control. In: Proc. 42nd IEEE Conference on Decision and Control, Maui, Hawaii, USA, December 2003, pp. 2374–2380 (2003)
5. Bhaya, A., Kaszkurewicz, E.: A control-theoretic approach to the design of zero finding numerical methods. *IEEE Transactions on Automatic Control* 52(6), 1014–1026 (2007)
6. Pazos, F., Bhaya, A., Kaszkurewicz, E.: Unified control Liapunov function based design of neural networks that aim at global minimization of nonconvex functions. In: Proc. of the International Joint Conference on Neural Networks, Atlanta, U.S.A. (2009)
7. Polyak, B.T.: Some methods of speeding up the convergence of iterative methods. *USSR Computational Mathematics and Mathematical Physics* 4(5), 1–17 (1964); *Zh. Vychisl. Mat. Mat. Fiz.*, pp. 791–803 (Russian edn.)
8. Polyak, B.T.: Introduction to optimization. Optimization software, New York (1987)
9. Attouch, H., Goudou, X., Redont, P.: The heavy ball with friction method. The continuous dynamical system. *Communications in contemporary math.* 2, 1–34 (2000)
10. Cabot, A.: Inertial gradient-like dynamical system controlled by a stabilizing term. *Journal of Optimization Theory and Applications* 120(2), 275–303 (2004)
11. Alvarez, F., Attouch, H., Bolte, J., Redont, P.: A second-order gradient-like dissipative dynamical system with Hessian-driven damping. Application to optimization and mechanics. *J. Math. Pures Appl.* 81, 747–779 (2002)
12. Bhaya, A., Kaszkurewicz, E.: Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method. *Neural Networks* 17(1), 65–71 (2004)
13. Cortés, J.: Discontinuous dynamical systems. A tutorial on notions of solutions, nonsmooth analysis, and stability. *IEEE Control systems magazine*, 36–73 (June 2008)

On the Knowledge Organization in Concept Formation: An Exploratory Cognitive Modeling Study

Toshihiko Matsuka¹, Hidehito Honda¹, Arieta Chouchourelou²,
and Sachiko Kiyokawa³

¹ Department of Cognitive and Information Science, Chiba University, Japan

² European University Cyprus, Nicocia, Cyprus

³ Department of Psychology, Chubu University, Japan

Abstract. Recent cognitive modeling studies suggest the effectiveness of metaheuristic optimization in describing human cognitive behaviors. Such models are built on the basis of population-based algorithm (e.g., genetic algorithm) and thus hold multiple solutions or notions. There are, however, important yet unaddressed issues in cognitive mechanisms associated with possession of multiple notions. The issues we address in the present research is about how multiple notions are organized in our mind. In particular, we paid close attention to how each notion interact with other notions while learning a new concept. In so doing, we incorporated Particle Swarm Optimization in a cognitive model of concept learning. Three PSO-based concept learning models were developed and compared in the present exploratory cognitive modeling study.

1 Introduction

Human cognitive behaviors are generally considered as rational (e.g. [1]). There is a school of thought which claims that rational probabilistic reasoning generates such behaviors [1], while a different school claims that humans' rationality is bounded and those behaviors are driven by rather simple heuristics [4]. Computational models of human cognition based on the former principle employ, for example, bayesian statistics and derivative-based optimization methods (e.g. [1]). A majority of cognitive models is built on the basis of this principle, and we refer to this type of cognitive models as *rationality* models (RM hereafter). A smaller number of cognitive models are built on the basis of the latter principle and those models often employ some form of metaheuristic optimization methods [9]. We refer to this type as *bounded rationality* models [4] or simply BRM.

There are several differences between those two types of models. One important and significant difference is that while the vast majority of RMs searches for and holds a single optimal solution (i.e., optimizing a single set of coefficients), BRMs would obtain multiple, often diverse, “optimal” solutions (i.e., optimizing multiple sets of coefficients). This difference is critical as that the results of human laboratory experiments (e.g. [2]) suggest that humans indeed hold multiple solutions or notions (Note that we refer to a single complete set of model coefficients as a self-sufficient notion in this paper). In addition, human learning is believed to involve combining ideas and hypotheses. In order for a cognitive model to integrate this “learning-by-combination”

mechanism, it needs to possess more than one notion. Moreover, the capability of maintaining multiple notions permits a model to exhibit very adaptive behaviors by selecting a situationally appropriate notion to manifest from a pool of notions [9]. This uni-vs-multiple notion issue is not inherent to the difference in the level of rationality (i.e., the multi-notion mechanism can be integrated into RM, and BRM can be modeled with a single notion). Rather, the difference is largely determined by the typical optimization method applied to RM (gradient descent) and BRM (evolutionary computation). Nonetheless, BRMs with meta-heuristic optimization methods demonstrate a greater level of descriptive validity with regards to the multi-notion mechanism.

There are, however, important yet unaddressed issues in cognitive mechanisms associated with possession of multiple notions. The issues we address in the present research is about how multiple notions are organized in our mind. In particular, we paid close attention to how each notion interacts with other notions while learning a new concept. In so doing, we incorporated Particle Swarm Optimization (PSO) in a cognitive model of concept learning. PSO is a type of meta-heuristic optimization methods built on the basis of the principle of collective behaviors exhibited by simple organisms [3] [6]. Its fundamental principle follows simplified psycho-social behaviors; each quasi-freely acting organism within a society interacts and exchanges information with other organisms in order for the society as a whole to arrive or realize “optimal” solutions. More specifically, in PSO each organism moves within a multidimensional coefficient space, where the movement (i.e., direction and distance) is determined by both personal experience (i.e., the best solution that each organism has found) and social knowledge (i.e., the best solution shared by a group of organisms).

One effective feature of PSO in modeling human concept learning is that PSO allows us to manipulate its social structures that controls flow of information and thus behaviors of optimization processes. This in turn, allows us to examine and compare various types of knowledge organizations in cognitive models in order to identify descriptively “realistic” knowledge organizations. This aspect is important as that most, if not all, existing multi-notion BRMs incorporate a simple knowledge organization (e.g. there is only one knowledge cluster and all notions are connected with each other). Thus, the influence of knowledge organization in concept formation has not been systematically examined in the past. In the present paper, we explore three types of knowledge organization in concept formation in order to examine influence of knowledge organization in learning a new concept.

2 Particle Swarm Optimization (PSO)

2.1 Standard PSO

The fundamental computational operation in PSO is simple. In PSO, each potential solution (i.e., single particle or notion) is quasi-freely moving with some velocity in a hyper-dimension coefficient space. The position of a notion at time $t + 1$ is determined by adding its velocity to the current (i.e., t) position. Thus,

$$\theta_i^{(j)}(t + 1) = \theta_i^{(j)}(t) + v_i^{(j)}(t + 1) \quad (1)$$

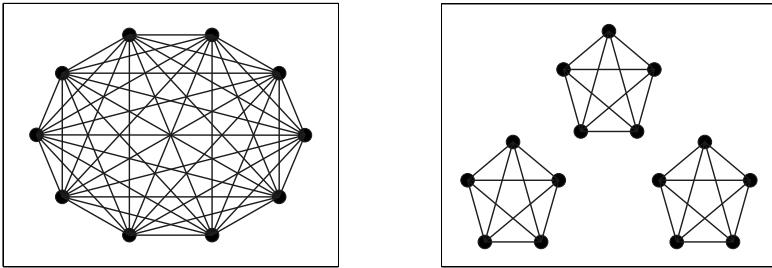


Fig. 1. Social structures of standard PSO. Left panel: Uni-cluster PSO where all notions are connected and exchange information with each other. Right panel: Multi-cluster PSO. All notions within a given cluster C_k are connected, but there is no connection between different clusters. Each cluster is independent of each other in Multi-cluster PSO. Note that these figures only show social structures and are not plotted on the real axes.

where θ s are model coefficients, i indicates coefficients and j indicates particles or notions. The velocity of a particle is determined by both its personal experience in its memory trace and the socially exchanged information. The personal experience simply refers to the position that resulted in the optimal solution for each particle (i.e., notion). The socially exchanged information refers to the position that resulted in the optimal solution among a group of particles. Specifically, the velocity for notion j at time $t + 1$ is given as:

$$v_i^{(j)}(t + 1) = v_i^{(j)}(t) + \omega_1 U_{i1} \left(Pb_i^{(j)} - \theta_i^{(j)} \right) + \omega_2 U_{i2} \left(Gb_i^{(C_k)} - \theta_i^{(j)} \right) \quad (2)$$

where U_i is a random number drawn from the Uniform distribution, $Pb_i^{(j)}$ indicates the best position (i.e., the best solution) for notion j for coefficient i , and the $Gb_i^{(C_k)}$ indicates the best position shared among a group of notions that belongs to knowledge cluster k . ω s are scalers weighting relative importance of personal experience (i.e. personal best solution) and social information (i.e., global best solution). The positions of notions are iteratively updated until some stopping criterion is met.

Model Interpretation. In PSO, each notion has its own tendency in knowledge modification (i.e., $v_i^{(j)}$ & $Pb_i^{(j)}$). In addition, PSO capitalizes opportunity of possessing multiple notions by combining notions (i.e., $Gb_i^{(C_k)}$) in learning. When ω_{i2} is sufficiently large relative to ω_{i1} , then initially diverse notions in a cluster k gradually converge to the group's best solution, $Gb_i^{(C_k)}$. This can be interpreted as that humans initially create sets of random hypotheses and those hypotheses become crystalized as learning progresses. Conversely, when ω_{i2} is sufficiently small, then initially diverse notions in knowledge cluster k would remain diverse after learning. If we assume a single notion cluster (Fig. 1, left panel) as in the majority of existing BRMs, then the effect of the global best solution need to be reduced in order to obtain diverse notions, which is empirically observed in some behavioral studies. [2] [7]. However, we find reducing the global effect normatively and descriptively questionable. It is normatively inefficient,

because it simply becomes a series of independent stochastic optimizations when information is not shared among the notions. It sounds descriptively invalid, because that (1) it is simply unrealistic to assume that humans do not emphasize the best available notion in learning a concept, and that (2) a series of independent stochastic optimization requires an excessive amount of cognitive effort.

Alternatively, if we assume multiple hypothesis clusters (Fig. 1, right panel), then it can obtain multiple diverse notions without reducing the effect of the global best solutions in learning. In the present paper, in order to evaluate this supposition, we compare learning characteristics of Uni-cluster PSO (UP) and Multi-cluster PSO (MP) using a hypothetical concept learning task.

2.2 Shotgun PSO

In addition to UP and MP, we also examined Shotgun PSO (SP). In Shotgun PSO, there is a small number of focal notions (as few as the number of clusters) and those focal notions are aided by auxiliary notions (or hypotheses) that float around the focal notions. A focal notion is analogous to a target (or bull's eye) in a shotgun shooting, and auxiliary notions are analogous to bullets (Fig. 2). Operationally, auxiliary notions are generated by adding random noise to the focal notions [5]. Specifically, an auxiliary notion, ξ at time $t + 1$ is given as:

$$\xi_i^{(j \in \mathcal{C}_k)}(t + 1) = \theta_i^{(\mathcal{C}_k)}(t) + N_i(0, \sigma_i^{(j \in \mathcal{C}_k)}(t + 1)) \quad (3)$$

where θ is a focal notion, N_i is a function that returns a random number drawn from the Normal distribution with the corresponding parameters. The standard deviations, σ_i are given as:

$$\sigma_i^{(j \in \mathcal{C}_k)}(t + 1) = \eta \sigma_i^{(j \in \mathcal{C}_k)}(t) \cdot \exp(N_G(0, \gamma_g) + N_i(0, \gamma_i)) \quad (4)$$

where $0 < \eta < 1$ is a scalar that controls the rate of decay in σ s, N s are as in Eq. 3, and γ s define thoroughness or search widths. Assuming that learning is a minimization problem, Shotgun PSO then tries to identify the best auxiliary notion (i.e., Gg).

$$Gb_i^{(\mathcal{C}_k)} = \xi_i^{(j)} : Y(\xi_i^{(j)}) \leq Y(\xi_i^{(m)}), \forall \{j \neq m\} \in \mathcal{C}_k \quad (5)$$

where Y is a function that evaluate utility of notions (Note that learning is assumed as a minimization problem). Once the best auxiliary notion is identified, then the focal notion for cluster k , $\theta_i^{(\mathcal{C}_k)}$ is updated as follows:

$$\theta_i^{(\mathcal{C}_k)}(t + 1) = \theta_i^{(\mathcal{C}_k)}(t) + v_i^{(\mathcal{C}_k)}(t + 1) \quad (6)$$

$$v_i^{(\mathcal{C}_k)}(t + 1) = v_i^{(\mathcal{C}_k)}(t) + \omega_1 U_{i1} \left(Pb_i^{(\mathcal{C}_k)} - \theta_i^{(\mathcal{C}_k)} \right) + \omega_2 U_{i2} \left(Gb_i^{(\mathcal{C}_k)} - \theta_i^{(j)} \right) \quad (7)$$

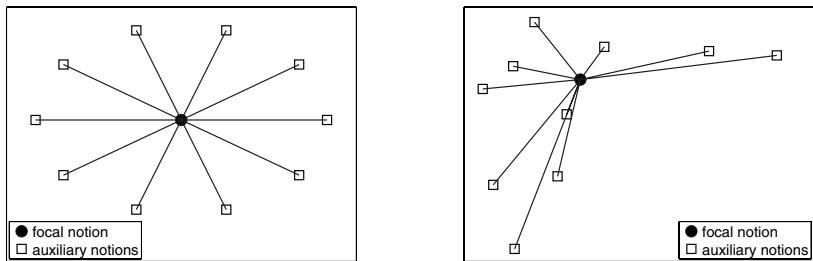


Fig. 2. Knowledge structures of Shotgun PSO. Left panel shows the social structure of Shotgun PSO and are not plotted on the real axes. All auxiliary notions within a cluster are connected with the focal notion, but there is no connection among auxiliary notions. Right panel shows social structure of Shotgun PSO plotted on a hypothetical 2-dimensional coefficient space.

Remarks on Shotgun PSO. In both Uni-notion PSO (UP) and Multi-notion PSO (MP), each notion within a cluster was connected to and exchanges information with all other notions within the same cluster. This can be interpreted as that all notions in our mind have equal contributions and weights in learning. However this implicit assumption becomes unrealistic when the number of notions becomes large for a give task. In this regards, Shotgun PSO offers better model interpretation. In Shotgun PSO, humans possess small numbers of independent focal notions (as small as the number of knowledge clusters) and they generate sets of hypotheses on the basis of those focal notions. Most of those hypotheses are short-lived or are forgotten unless they are found to be “useful.” If a hypothesis is determined as useful, then it would be incorporated into the existing notions.

As in other population-based metaheuristic optimization methods, many notions generally become very similar to each other in latter stages of learning in both UP and MP. It is, however, unrealistic that humans maintain and hold very similar notions separately. In Shotgun PSO, the number of meaningful notions (i.e., focal notion) is kept small throughout learning, and it is less likely than UP and MP to result in acquiring useless closely-packed notions that just take up memory resources without providing any additional insights.

3 Simulation

When we think, reason, and communicate, we have a tendency to use knowledge at a high level of abstraction. Although we may lose some information through abstraction, the use of categorical knowledge is essential. By compressing the vast amount of available information, a cognitive process called *categorization* allows us to process, understand, and communicate complex thoughts and ideas by efficiently utilizing salient and relevant information while ignoring other types of information. Learning new categories is thus equivalent to learning concepts. In the present research, we simulate a hypothetical category learning tasks in order to compare 3 types of PSO.

3.1 Modeling Concept Formation – Category Learning

3.2 Categorization Model

We used SUPERSET [8] framework, which is essentially RBF with Mahalanobis distance, for modeling categorization (i.e., forward) processes. The knowledge architecture of SUPERSET is built on the basis of a superset of three main theories of categorization, exemplar, prototype, and rule theories. SUPERSET assumes that humans utilize similarities between the input stimulus and internal (memorized) reference points (i.e., rules, prototypes, or exemplars) as evidence to probabilistically assign the input stimulus to an appropriate category. The psychological similarity (i.e., s_j) between the j th reference point, r_j , and an input stimulus, x , is determined by the Mahalanobis distance between them.¹ Note that stimuli are defined by I -dimensional space. The following exponential function is used to estimate the psychological similarity:

$$s_j^{(n)}(\mathbf{x}) = \exp\left(-\beta \left[\left(\mathbf{r}_j^{(n)} - \mathbf{x} \right)^T \mathbf{A}_j^{(n)} \left(\mathbf{r}_j^{(n)} - \mathbf{x} \right) \right]\right) \quad (8)$$

where β is a scaling parameter defining the overall similarity gradient, superscript n indicate different strategies (i.e., individual notions), and \mathbf{A}_j is $I \times I$ selective attention matrix for reference point j that transform physical distances into psychological ones. Elements of \mathbf{A}_j have the following properties: $0 \leq a_{ii} \leq 1$; $a_{im} = a_{mi}$; $-1 \leq a_{im} \leq 1$, $\forall i \neq m$; $|a_{im}| \leq \sqrt{a_{ii}a_{mm}}$. The diagonal entries are attention weights that scale individual feature dimensions and the off diagonal entries are attention weights for covariations between feature dimensions.

The similarity-based activations are then fed forward to the k -th output node (e.g., output for category k), O_k , weighted by w_{kj} , which determines the strength of association between each reference point j and each output node k :

$$O_k^{(n)}(\mathbf{x}) = \sum_j w_{kj}^{(n)} s_j^{(n)} \quad (9)$$

The probability of categorizing input instance x to category C is based on the activation of output node C relative to the activations of all output nodes:

$$P(C|\mathbf{x}) = \frac{\exp\left(\phi \cdot O_c^{(\nu)}(\mathbf{x})\right)}{\sum_k \exp\left(\phi \cdot O_k^{(\nu)}(\mathbf{x})\right)}. \quad (10)$$

where ϕ controls decisiveness of the classification response, and the superscript ν indicates the notion adopted to make a categorization response (i.e., a particle that is located at the best position).

3.3 Knowledge Utility Estimation

We assume what constitutes the “best” or “optimal” solution is situational specific. Real-world human learning is more likely to be explained as a process of multi-objective

¹ Subscripts and superscript used in describing SUPERSET and those for PSO do not correspond with each other.

optimization of utility of knowledge, where objectives are not only defined by categorization errors, but also by each learner's internal and external factors. Accordingly, our PSO embedded SUPERSET assumes that learning is driven by an optimization of the subjectively and contextually defined utility of knowledge being acquired, rather than by a simple classification error minimization routine. For the sake of simplicity, we assume that learning involves only error and complexity minimizations in the present simulation. Thus,

$$Y(\mathbf{w}^{(n)}, \mathbf{a}^{(n)}) = E(\mathbf{w}^{(n)}, \mathbf{a}^{(n)}) + \lambda_w \sum_k \sum_j (w_{kj}^{(n)})^2 \\ + \lambda_{a1} \sum_i \left[1 + (a_{ii}^{(n)})^{-2} \cdot \sum_l (a_{ll}^{(n)})^2 \right]^{-1} + \lambda_{a2} \sum_j \sum_i \sum_{m=i+1}^{I-1} (a_{jm}^{(n)})^2 \quad (11)$$

The first term corresponds to categorization error. The second, third, and forth terms correspond to notion complexity, where the second term defines notion complexity based on association weights, third for dimensional attention weights, and forth for correlational attention weights [8].

The predicted (in)accuracy of a notion during categorization is estimated based on a retrospective verification function [9], which assumes that humans estimate the accuracies of the notions by applying the current notions to previously encountered instances with a memory decay mechanism. The retrospective verification function operates like locally weighted regression, paying more attention to “applicabilities” or fitness in recent training instances than older ones.

$$E(\theta^n) = \sum_{t=1}^T \left[\pi(t) \sum_k^K \left[d_k^{(t)} - O_k^{(n)}(x^{(t)}) \right]^2 \right] \quad (12)$$

where t indicates time, and the last term is the sum of squared error with d_k being the desired output for category k . $\pi(t)$ is the (training) exemplar retention weight that defines the strength of the retaining training exemplar $x^{(t)}$ (and the corresponding desired output).

3.4 Methods

There were three types of PSO embedded SUPERSET learners involved in the present simulation study, namely UP(Uni-cluster PSO) who maintains a single cluster of notions, MP (Multi-cluster PSO) who maintains multiple clusters of notions, SP (Shotgun PSO) who maintains a small number of focal notions. All models had the identical forward algorithm (i.e., Eqs. 8-10), and utility estimation function (i.e., Eqs. 11 & 12).

All models were run in a simulated training procedure to learn the correct classification responses for the stimuli with corrective feedback. There were a total of 100 training blocks, each of which was organized as a random presentation of 9 unique exemplars in Table II. The model parameters were selected arbitrarily; $c = 3$, $\phi = 5$, $\lambda_w = 0.01$, $\lambda_{a1} = 1$, $\lambda_{a2} = 1$, $\omega_1 = 0.3$. The parameter that controls the level of influence of social information, ω_2 were randomly drawn from the Uniform distribution. The total numbers of notions for UP and MP were 30, and the numbers of clusters were 1

Table 1. Schematic representation of stimulus set used in Simulation Study

		Stimulus Set			
Category		Dim1	Dim2	Dim3	Dim4
A	1	1	1	1	1
A	1	1	2	2	
A	1	1	3	3	
B	2	2	4	1	
B	2	2	5	2	
B	2	2	6	3	
C	3	3	7	1	
C	3	3	8	2	
C	3	3	9	3	

and 3 for UP and MP, respectively. The number of focal notions for SP was fixed at 3, and each focal notion had 10 auxiliary notions. There were a total of 1000 simulated participants per each model.

Stimuli. Table 1 shows schematic representation of stimulus set used in simulation. Note that Dimensions 1 and 2 are redundant and are also perfectly correlated with the category membership, each being a necessary and sufficient diagnostic dimension. All feature values in Dimension 3 were treated as nominal values differentiating each element within the dimension, and thus their numeric differences do not have any meaning. Dimension 4 is an irrelevant dimension containing useless information for categorization. Dimensions 1 and 2 provide information on a common component shared by members within categories while Dimension 3 serves as an idiosyncratic component that differs from instance to instance within and also across categories. Thus, information at either or both the category (i.e., Dim1 & Dim2) and exemplar (Dim3) levels would allow categorization of the stimuli. Note that this stimulus set is very simple to categorize and that there are several valid categorization strategies. An empirical study [7] that used stimulus set similar to ours showed that humans chose to pay attention primarily if not exclusively to either one of the solely diagnostic dimensions. Another interesting phenomenon reported in that study [7] is that some participants who learned to pay attention to either dimension reported that they realized that there was another diagnostic dimension, indicating the possibility of possessing multiple notions or solutions.

In the present study, we paid attention to the numbers of “distinctive” useful notions acquired by the three PSO embedded SUPERSET. In so doing, we counted the numbers of notions that met all of the following conditions: (1) categorization accuracies were greater than 0.85, (2) more than three forth of the total amounts of attention were paid to a single dimension. We did not count notions if there were similar notions that were already counted as distinctive useful notions.

3.5 Results and Discussion

Table 2 shows the results of the simulation study. All models successfully learned to categorize the stimuli (accuracies being 0.91, 0.95, 0.95 for UP, MP, and SP respectively).

Table 2. Results of Simulation

	Uni-cluster.	Multi-Cluster	Shotgun
Accuracy	0.91	0.95	0.95
No. distinctive notions	1.19	2.33	2.25

There were some cases of unsuccessful learning in all models. Those cases were largely caused by extreme ω_2 values (i.e., effect of social information), which were randomly drawn in the simulation. Thus learning was less likely to diverge with extreme ω_2 values in MP and SP than UP. This turn indicates that learning is more robust in MP and SP than UP.

In addition, there was a clear difference between UP and the remaining two models in the numbers of distinctive notions acquired during learning. On average UP acquired only one notion, while MP and SP acquired two notions. Although these two models did not explicitly search for diverse notions, it successfully acquired, on average, more than two distinctive accurate notions. Given that several empirical studies (e.g. [2], [7]) showed that humans often hold multiple notions or strategies for a given task, MP and SP's predictions were more consistent with real human cognitive behavior than that of UP (UP probably could acquire multiple notions if its learning objective function contained some form of "diversity") There was, however, no clear difference between MP and SP. The categorization accuracies for and the number of distinctive notions acquired by MP and SP were almost identical.

Although MP and SP exhibited similar cognitive behaviors, we found model interpretation of SP more descriptively valid as noted above. In addition, given that SP requires to hold much smaller numbers of notions (3 vs. 30 in simulation), cognitive effort required or modeled by SP is much more feasible than MP. The results can be interpreted as that SP had acquired as many distinctive notions as MP with much smaller cognitive efforts. In other words, MP had acquired multiple effective notions, but there was a great degree of redundancies and duplications.

4 Conclusion

Recent cognitive modeling studies suggest the effectiveness of meta-heuristic optimization in describing human cognitive behaviors. Such models are built on the basis of population-based algorithm (e.g., genetic algorithm) and thus hold multiple solutions or notions. There are, however, important yet unaddressed issues in cognitive mechanisms associated with possession of multiple notions. The issues we address in the present research was about how multiple notions are organized in our mind. In particular, we paid close attention to how each notion interact with other notions while learning a new concept. In so doing, we incorporated Particle Swarm Optimization (PSO) in a cognitive model of concept learning, because PSO allows us to manipulate its social structures that controls flow of information, which in turn, allows us to examine and compare various types of knowledge organizations. Three PSO-based concept learning models were developed and compared in the present exploratory cognitive modeling

study. The results of our simulation indicated that humans are more likely to have multiple independent clusters of notions in their minds, and that the clusters are driven by a small number of leading notions.

Acknowledgements

This work was in part supported by the Japan Society for the Promotion of Science KAKENHI (Grant No. 20700235) and Support Center for Advanced Telecommunications Technology Research (SCAT).

References

1. Anderson, J.R.: *The Adaptive Character of Thought*. Lawrence Erlbaum, Hillsdale (1990)
2. Anderson, R.C., Pichert, J.W.: Recall of previously unrecallable information following a shift in perspective. *Journal of Verbal Learning and Verbal Behavior* 17, 1–12 (1978)
3. Engelbrecht, A.P.: *Fundamentals of Computational Swarm Intelligence*. Wiley, Chichester (2005)
4. Gigerenzer, G., Todd, P.M., The ABC Research Group: Simple heuristics that make us smart. Oxford, New York (1999)
5. Higashi, H., Iba, H.: Particle Swarm Optimization with Gaussian Mutation. In: *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 72–79 (2003)
6. Kennedy, J., Eberhart, R.C., Shi, Y.: *Swarm Intelligence*. Morgan Kaufmann, San Francisco (2001)
7. Matsuka, T., Corder, J.E.: Observed attention allocation processes in category learning. *Quarterly Journal of Experimental Psychology* 61, 1067–1097 (2008)
8. Matsuka, T., Sakamoto, Y., Chouchourelou, A.: Modeling a flexible representation machinery of human concept learning. *Neural Networks* 21, 289–302 (2008)
9. Matsuka, T., Sakamoto, Y., Chouchourelou, A., Nickerson, J.V.: Toward a descriptive cognitive model of human learning. *Neurocomputing* 71, 2446–2455 (2008)

Dynamics of Incremental Learning by VSF-Network

Yoshitsugu Kakemoto¹ and Shinchi Nakasuka²

¹ The JSOL, Ltd, Tokyo, Harumi Center Building, 2-5-24 Harumi, Chuo-ku Japan
`kakemoto.yoshitsugu@jsol.co.jp`

² The University of Tokyo, Tokyo, 7-3-1 Hongo, Bunkyo-ku Japan
`nakasuka@space.t.u-tokyo.ac.jp`

Abstract. In this paper, we report the dynamics of VSF-Network. VSF-Network is a neural network for the incremental learning and it is a hybrid neural network combining the chaos neural network with a hierarchical network. VSF-Network can find the unknown elements from input with clusters generated by the chaos neuron. We introduce new incremental learning model to explain the dynamics of VSF-Network in this paper. We show the result of analysis of the dynamics of VSF-Network. In the analysis, we focused on the connection weights between layers and neuron cluster generated by the chaotic behavior.

Keywords: Complex system, Chaos neural network, Knowledge acquisition, Nonlinear dynamics.

1 Introduction

Symbol is fundamental for our cognitive abilities. We come to realize and comprehend language through a series of semantic interpretations of symbols and meanings within our world[1]. For the symbol acquisition, the various models have been proposed. The core of those is the abstraction of external stimuli and the grounding of symbol. The abstraction of stimuli is an instance of the incremental learning[2] and it has the special properties. If we implement the abstraction process by neural networks, the patterns generated by the abstraction should have the following properties.

- A neural network saves its connection weights learned in the prior learning and it learns only the new elements in the patterns given at the successive learning.
- The elements in patterns are learned by a neural network independently, if they are independent. If we can divide the elements in patterns into independent parts, a pattern which consists of each part can be recognized. If elements in the pattern are composed from others, every element can be recognized only by the learned pattern that composes the whole pattern.

We call this incremental learning the additive learning[3]. For the additive leaning, we have proposed VSF-Network, Vibration Synchronizing Function Network[3].

VSF-Network is a hybrid neural network with the combination of hierarchical neural network(HNN) and chaotic Neural Network(CNN)[4]. The CNN has the ability to determine whether an input pattern is known one or unknown one[5]. Using the CNN to the classifier for know or unknown pattern, we can develop an autonomously-controlled incremental learning.

2 Additive Learning by Neural Network

2.1 Incremental Learning

The incremental learning on HNN with the back-propagation suffers from forgetting. One way to implement the incremental learning is to store the independent elements in patterns into sub-networks of a neural network. In many cases, a neural network has redundant neurons after learning. We can use these neurons to learn new elements in a pattern. If a neural network can find the new elements and can learn them by an independent sub-network, we can implement the first step of the incremental learning.

2.2 Chaotic Neural Network and Pattern Selection

We employ the CNN to find unknown or known elements in a pattern. The dynamics of the i^{th} Chaotic Neuron(CN) in a neural network composed of M CN can be modeled as,

$$x_i(t+1) = f[\xi_i(t+1) + \eta_i(t+1) + \zeta_i(t+1)], \quad (1)$$

In (1), $x_i(t+1)$ is the state of the i^{th} CN at the discrete time $t+1$, f is the output function and each term is defined as follows.

$$\xi_i(t+1) = k_s \xi_i(t) + \sum_{j=1}^M v_{jk} A_j(t) \quad (2)$$

$$\eta_i(t+1) = k_n \eta_i(t) + \sum_{j=1}^N w_{ij} x_j(t) \quad (3)$$

$$\zeta_i(t+1) = k_r \zeta_i(t+1) - \alpha x_i(t) - \theta(1 - k_r). \quad (4)$$

In (2), $A(t)$ is input at time t and v_{jk} is the connection weight between the k^{th} element of input pattern and the CN j . In (3), w_{ij} is the connection weight between CN i and CN j . k_s , k_n and k_r is a parameter for the each term. In (4), α is the parameter and θ is the threshold.

The CNN has an ability to find an unknown pattern for the network[5]. If an input pattern is known for the network, the state of the CNN will be stable. If an input pattern is unknown for the network, the state of the CNN can not reach a stable state because of the chaotic itinerancy.

2.3 Detection of Neuron Group

By using the CNN, we can detect an unknown pattern, but we can not identify which element of a pattern is unknown. We want to find unknown elements and known element in a pattern.

To bring out the relation between the behavior of each element and the property of the pattern, we analyze the CNN based on the GCM model (Globally Coupled Map [6]). GCM is a model of a system that nonlinear elements are connected to each other, and all elements are influenced through total amount brought from each element. The CNN is an instance of GCM[7].

We define a synchronized state that an output from the element i is coincident with the output from the other element j . The element group of synchronized elements is called a cluster. The factor of clustering is the structure of the manifold spanned by the GCM[8]. The manifold consists of three different manifold, the stable manifold E_s , the unstable manifold E_u and the center manifold E_c . If chaotic strength of each element is weak, several elements are trapped in an attractor on E_u , other elements are not trapped in any attractor and they show itinerancy attracting among these manifold. After the retrieving, the CNN returns the clustering information for the neurons. We use the information to find the known elements in a pattern. The elements in a cluster corresponding to a known part in a pattern and other elements which are not clustered correspond to an unknown part in a pattern.

We apply the output from the hidden layer neuron of HNN to the input of the CNN. After a learning phase, HNN composes a solution space. If this solution space corresponds to the manifolds on GCM, this solution space has a stable subspace and an unstable subspace. We assume that there are basins corresponding to the learning result in the stable subspace and the learned elements in the pattern are easily trapped in the basins. By the trapping, the state of the element falls in a stable state. In contrast, the unknown elements in a pattern are not trapped in any basin.

3 VSF-Network

We have proposed VSF-Network[3]. We reconsidered its dynamics and configuration to improve its ability for the incremental learning. In Fig 1, an overview of VSF-Network is shown. VSF-Network is composed of BP-module that is a hierarchical network trained with the selective back-propagation and CNN-module that finds known or unknown parts of a pattern.

The learning of VSF-Network is assumed that the connection weights between each layer have been acquired before the learning. The learning of VSF-Network is performed as follows.

1. Patterns are input into the input layer of BP-module.
2. The outputs from the hidden layer in BP-module sent to CNN-module and they are used for the initial state of each neuron of CNN-module.

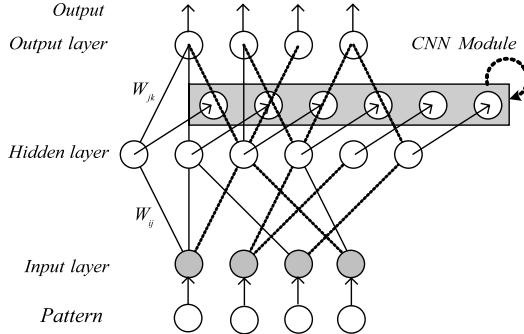


Fig. 1. An Overview of VSF-Network

3. From the initial state, CNN-module performs the retrieving based on the dynamics of CNN-module (1) for times $t = 1, \dots, T$. The consistent rates (5) between neurons in CNN-module are calculated.
4. The rest process of the forward path is performed. The error between the output from BP-module and the correct output for the pattern is calculated.
5. The connection weights between each layer are updated based on the weight update rule (7).

3.1 Pattern Detection at CNN-Module

After the retrieving, the consistent rate between all neurons of CNN-module is calculated. This consistent rate is the basis for the determining whether a neuron i belongs to a cluster or not. If the consistent rate for a neuron is high, the neurons in a cluster and they are assumed to be stable. A neuron whose consistent rate stays low is not in any cluster and it is assumed to be unstable. The consistent rate $\lambda_{ij}(t)$ between a CN i and other CN j at time t is defined as,

$$\lambda_{ij}(t) = \frac{1}{T} \sum_{n=t}^{t+T} C_n(\varepsilon^{spl}, x_i, x_j). \quad (5)$$

In (5), $C_n(\varepsilon^{spl}, x_i, x_j)$ is the function for the consistent between the state x_i of the CN i and the state x_j of the CN j . ε^{spl} is a constant to define a separation between the state x_i and the state x_j and H is the Heaviside function.

$$C_n(\varepsilon^{spl}, x_i, x_j) = H(\varepsilon^{spl} - |x_i - x_j|) \quad (6)$$

3.2 Weight Update Rule

The connection weights between each layer of BP-module are updated based on the result on CNN-module. In (7), λ_i is the mean of the consistent rate calculated for the CN i to other CNs. If the λ_i of the CN i is less than a constant P , all

connection weights for the CN i are updated. If the λ_i of the CN i is not less than a constant P , all connection weights for the CN i are not updated.

In addition to the clustering information, we introduce a geometrical relation between each subspace into the weight update rule. The subspace is a part of the manifold spanned by the connection weights of VSF-Network. If a input pattern is an unknown pattern, the network learns it using an unstable subspace, but there is many candidate subspaces for the learning. Therefore, we introduce an additional rule for selecting subspace. We assume that an unstable subspace having a less correlation to other subspace is redundant and we use this subspace to learning new input pattern. Based on this idea, we introduce the correlation of every connection weight between the input layer and the hidden layer. The weight update rule on VSF-network is defined as follows.

$$\Delta W_{jk} = \begin{cases} \eta \left(\frac{\partial E_{jk}}{\partial W_{jk}} \right) (\lambda_i \leq P) \\ 0 \quad (\lambda_i > P) \end{cases}, \Delta W_{ij} = \begin{cases} \eta \left(\frac{\partial E_{ij}}{\partial W_{ij}} \right) (\lambda_i \leq P) \\ 0 \quad (\lambda_i > P) \end{cases} \quad (7)$$

$$\frac{\partial E_{jk}}{\partial W_{jk}} = \prod_{i=1}^m (1.0 - |cor_{ij}|) \sum_{j=1}^n E^\mu f'(O_k^\mu) \quad (8)$$

$$\frac{\partial E_{ij}}{\partial W_{ij}} = (1.0 - |cor_{ij}|) \sum_{j=1}^n \frac{\partial E_{jk}}{\partial W_{jk}} f'(H_i^\mu) \quad (9)$$

In (7), η is a rate of update and E^μ is a MSE between the output from BP-module and the correct output for a pattern μ . H_j^μ is the output from neuron j in the hidden layer for the pattern μ . cor_{ij} is the correlation of connection weights between the hidden layer and the input layer for the hidden layer neuron i and j . f is the output function and it is the sigmoid function. O_k^μ is the output from the neuron k in the output layer for pattern μ .

4 Experiment

4.1 Experiment Condition

The experiments in this paper consist of the following three steps.

1. Pre-learning step: The step which the initial connection weights for VSF-Network is acquired.
2. Incremental learning step: At this step, we provide the patters differ from the patterns that are provided at the previous step.
3. Confirmation step: In this step, we provide all kinds of the patterns to examine the result of the learning at the previous steps.

4.2 Data for the Experiments

Input data used for experiments is distance data from a rover to obstacles or the outer circumference of the testing environment. The output to each input is whether the rover can avoid the obstacles if the rover turns to the left at the current position. The distance sensor is put on the center of the rover. The number of sensor beams is 26, and they are projected to the front of the rover but only 13 sensors on left side are used for the experiments because of simplifying calculation.

To generate total data set, we changed the size and the position of each obstacle. The position of the rover was also changed to generate the data set. All data are generated by a software simulator. The step of the experiments consisted of the number of trials. The input data for are randomly selected from the data set at every trial.

Three condition, simple obstacle, T-junction and composite obstacle are used for the experiment. Each obstacle condition is defined as follows.

1. T-junction: There is one obstacle at the left side of the rover.
2. Simple obstacle: There is one obstacle at the front of rover.
3. Composite obstacle: Both one simple obstacle and one T-junction appear on the experiment environment.

We assume that both the T-junction and the simple obstacle are independent obstacle condition each other. To verify this assumption, we calculate the correlation between data of the T-Junction and data of the simple obstacle. The coefficient of correlation is less than 0.2, so we assume that each condition is independent.

4.3 Parameter Settings

A parameter a in the output function for BP-module and CNN-module are 0.85. In the transfer function, a is 0.65. The defaults of parameters of VSF-network for the experiments are assigned as shown in Table 1. In particular, the values of parameters for CNN are assigned the CNN falls in a weak chaotic state. The number of neurons of VSF-network is assigned as shown in Table 2.

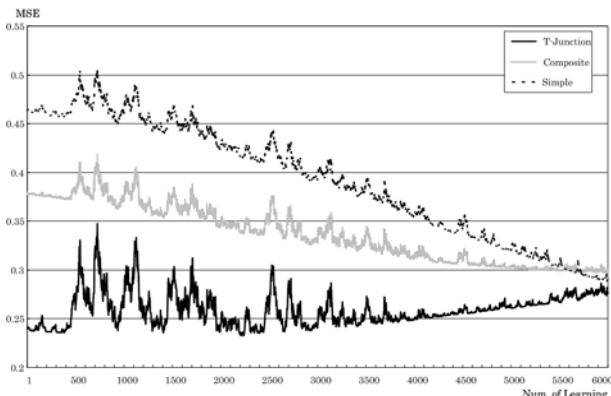
5 Dynamics on VSF-Network

5.1 Progress of Learning

We provide the T-junction obstacles in the pre-learning step and provided the simple obstacles task in the incremental learning step. At the confirmation step, we provide the T-junction obstacles, the simple obstacles and the composite obstacles to examine the results at the incremental learning step. The number of trials at the pre-learning step is 20,000 and the number of trials at the incremental learning step is 6000. The Mean Squared Error(MSE) of every learning improves with progress of the single obstacle learning as shown in Fig 2.

Table 1. Parameters and settings

Description	Value
d Temporal attenuation coefficient	5
k_s Parameter for the input term	0.562
k_n Parameter for the diffusion term	0.4774
k_r Parameter for the inhibitory term	0.2025
α Parameter for the inhibitory	2.6368
θ Threshold for inhibitory value	0.2437
ε^{spl} Parameter to determine a separation	0.001
η Parameter for weight update	0.1
T Length of measuring λ_{ij}	10
P Parameter to determine synchronous	0.48
Number of the input layer neuron in BP-module	13
Number of the hidden layer neuron in BP-module	26
Number of the output layer neuron in BP-module	1
Number of the input layer neuron in CNN-module	13

**Fig. 2.** Learning Progress by VSF-Network

By VSF-Network, the new input patterns are learned without forgetting the patterns learned in the previous learning step. VSF-Network outputs correct outputs for the single obstacles and for the composite obstacles. As for the compound obstacles, though any information about them is not provided during the both steps, the MSE for them improves. We can explain the improvement by the ensemble learning [9] [10], if we consider the sub-network corresponding to each obstacle that makes up the compound obstacle as weak learner. The compound obstacle can be recognized by their aggregation.

5.2 Know/Unknown Pattern Detection

In this section, we discuss about the internal dynamics of VSF-Network. At the first, we discuss a difference between behaviors on CNN-module for known and

unknown pattern. In Fig 3, we plot the state $x(t)$ of neurons in CNN-module on $x(t)-x(t+1)$ plane. The figure (1) and (2) in Fig 3 show $x(t)$ of the 7th neuron in CNN-module for an unknown pattern. The figure (3) and (4) in Fig 3 show $x(t)$ of the 11th neuron in CNN-module for a known pattern. The 7th hidden layer neuron weights the largest value to the 10th neuron in the input layer. The 11th hidden layer neuron weights the largest value to the 5th neuron in the input layer. The 5th input layer neuron catches the obstacle at the left side of the rover. The 11th input layer neuron catches the obstacle at the front of the rover. The figure (1) and (2) show $x(t)$ before the initial trial on the incremental learning step and the figure (3) and (4) show $x(t)$ after the final trial on the incremental learning step.

We calculate the Maximum Lyapunov Exponent(MLE) for each neuron in CNN-module. The 11th neuron shows the MLE less than 1.0 and the 7th neuron shows the MLE greater than 1.0 at the initial trial on the incremental learning step. The 7th neuron shows the MLE less than 1.0 at the final trial on the incremental learning.

These results suggest that unknown elements in a pattern do not fall in any basin on the stable subspaces. The neuron shows chaotic behavior on the retrieving at CNN-module. As a result, the connection weights are updated because the consistent rate of the neuron is low. By the weight update, new basin is created on an unstable subspace. Then, the neuron shows a stable behavior and the connection weights of the neuron are not updated any more.

The subspace for the 11th neuron on CNN-module and BP-module is stable and the subspace of the 7th neuron is unstable at the initial of the incremental learning, but it reaches an stable state at the end of the incremental learning

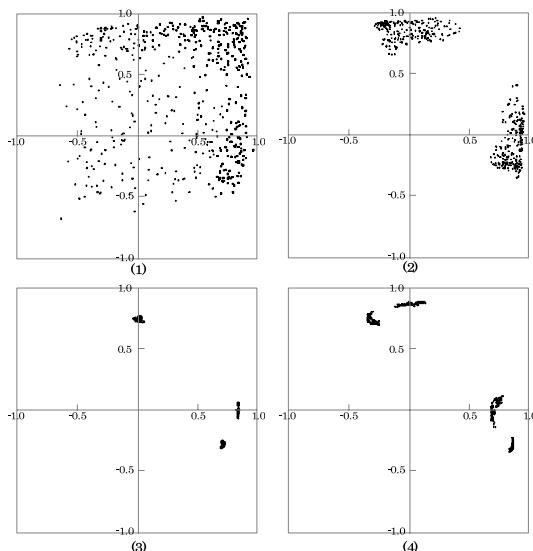


Fig. 3. State of neuron in CNN-module

step. The behavior of the 7th neuron shows a resistance for the noise. Its MLE do not change even if we add 20% of noise to the input data. The neurons in the hidden layer weight the neuron in the input layer which catches the left side obstacle show the same tendency.

5.3 Relation between Subspace and Pattern

Finally, we show the relation between the neuron in the hidden layer and the input layer. In Fig 4, the amounts of weight update for every neuron in the input layer in 300 trials are shown. The x axis of the figure is the number of learning and y axis of Fig 4 is the number assigned to every neuron in the input layer.

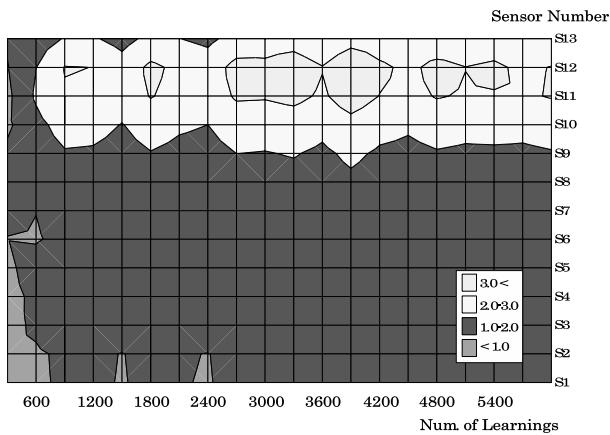


Fig. 4. Sum of update value for every neuron in the input layer

In the early stage of learning, each connection weight is updated with a small amount. After the 600 trials, update amounts for the 10th–12th neurons are increased, although the amounts for other neurons remain receiving a small amount. In the incremental learning phase, the 10th–12th neuron receive values from sensors which detect an obstacle in the front of the rover. An obstacle which is caught with the 10th–12th sensors is a unknown pattern for VSF-Network at the initial step of the incremental learning. Therefore, the connection weights corresponding to the sensors receive a large amount. Considering these result, the weight update rule works well in the incremental learning.

6 Conclusion

In this paper, we have shown an overview of VSF-Network and the discussion about the nonlinear dynamics on VSF-Network through experiments. For the incremental learning, VSF-Network shows good performance.

The ability of VSF-Network for the incremental learning is due to ability of the CNN. We can find the known and unknown elements in patterns by the CNN. In addition, we can identify a suitable subspace to learn new element in a pattern.

The next step of our research is the following two points. The first point is the examination about the relation between chaos strength of the CNN and dynamics of VSF-Network. The second point is consideration for generality of VSF-Network. It remains a possibility that the results of this paper are limited to the task in this paper.

References

1. Cangelosi, A., Hourdakis, E., Tikhonoff, V.: Language acquisition and symbol grounding transfer with neural networks and cognitive robots. In: International Joint Conference on Neural Networks, pp. 2885–2891 (2006)
2. Giraud-Carrier, C.: A note on the utility of incremental learning. *AI Communications* 13, 215–223 (2000)
3. Kakemoto, Y., Nakasuka, S.: The learning and dynamics of vsf-network. In: Proc. of ISIC 2006(2006)
4. Aihara, T., Tanabe, T., Toyoda, M.: Chaotic neural networks. *Phys. Lett.* 144A, 333–340 (1990)
5. Tanaka, T., Nakagawa, M.: A study of associative model with chaos neural network. *IEICE technical report. Neurocomputing* 95(57) (1995)
6. Kaneko, K.: Chaotic but regular pos-nega switch among coded attractors by cluster size variation. *Phys. Rev. Lett.* 63, 219 (1989)
7. Nozawa, H.: A neural network model as a globally coupled map ans applications based on chaos. *CHAOS* 2(2), 377–386 (1992)
8. Abe, S.: Chaotic itinerancy on 10gcm. RIMS Kokyuroku of Kyoto University 1244, 1–7 (2002)
9. Breiman, L.: Bagging predictors. *Machine Learning* (24), 123–140 (1996)
10. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1), 119–139 (1997)

Kernel CMAC with Reduced Memory Complexity

Gábor Horváth and Kristóf Gáti

Budapest University of Technology and Economics,
Department of Measurement and Information Systems
Magyar tudósok krt. 2. Budapest, Hungary H-1117

{horvath,gati}@mit.bme.hu

<http://www.mit.bme.hu>

Abstract. Cerebellar Model Articulation Controller (CMAC) has some attractive features: fast learning capability and the possibility of efficient digital hardware implementation. Besides these attractive features it has a serious drawback: its memory complexity may be very large. In multidimensional case this may be so large that practically it cannot be implemented. To reduce memory complexity several different approaches were suggested so far. Although these approaches may greatly reduce memory complexity we have to pay a price for this complexity reduction. Either both modelling and generalization capabilities are deteriorated, or the training process will be much more complicated. This paper proposes a new approach of complexity reduction, where properly constructed hash-coding is combined with regularized kernel representation. The proposed version exploits the benefits of kernel representation and the complexity reduction effect of hash-coding, while smoothing regularization helps to reduce the performance degradation.

Keywords: CMAC, kernel machines, smoothing regularization, hash-coding, memory complexity reduction.

1 Introduction

CMAC, a special neural architecture [1] has some attractive features. The most important ones are its extremely fast learning capability [2], and the special architecture that lets effective digital hardware implementation possible [3], [4]. The price of the advantageous properties is that its modelling capability is inferior to that of an MLP. This especially true for multidimensional case, as a multidimensional CMAC can approximate well only a function belonging to the additive function set [5]. A further deficiency of CMAC is that its generalization capability is also inferior to that of an MLP even for one-dimensional cases. This drawback was discussed in [6], where the real reason of this property was discovered and a modified training algorithm using smoothing regularization was proposed for improving the generalization capability. However, these modifications do not rectify the perhaps most serious drawback of CMACs: their extremely large memory complexity. Memory complexity, the size of weight memory of a

CMAC - especially in multidimensional cases - may be prohibitively large. Memory complexity problem had already been discussed in many papers. In the original Albus paper [1] hash-coding was proposed. Further studies [7], [8], however, showed that although hash-coding can reduce greatly the memory complexity of the network, it also deteriorates the performance: the generalization capability of a CMAC with hash-coding may be significantly reduced. Several works were dealing with further possibilities of complexity reduction. Among them one can find modular architectures, when a multidimensional CMAC is formed from many one- or two-dimensional CMAC modules. In these cases the results of these modules are aggregated to get the final result. The most important modular solutions are the sum of product (SOP) architecture [9], the hierarchical CMAC [10] or the MS-CMAC [11]. Although memory complexity of all these modular CMAC-variants is significantly reduced, the complexity of the resulted networks is still rather high. A further common drawback of all these solutions is that their training will be more complex. In [12] a kernel representation version was suggested, which when combined with a smoothing regularization reduces memory complexity, and at the same time it also improves modeling and generalization capability. This paper goes further on this road: now the combined application of regularized kernel representation and hash-coding is proposed, where regularized kernel representation may compensate the unfavorable effects of hash-coding.

2 A Short Overview of the CMAC

CMAC is an associative memory type neural network, which performs two subsequent mappings. The first one - which is a nonlinear mapping - projects an input space point $x \in R^N$ into an M -bit sparse binary association vector \mathbf{a} which has only $C << M$ active elements: C bits of the association vector are ones and the others are zeros. The second mapping calculates the output $y \in R$ of the network as a scalar product of the association vector \mathbf{a} and the weight vector \mathbf{w} :

$$y(\mathbf{x}) = \mathbf{a}(\mathbf{x})^T \mathbf{w} \quad (1)$$

CMAC uses quantized inputs and there is a one-to-one mapping between the discrete input data and the association vectors, i.e. each possible input point has a unique association vector representation.

Another interpretation can also be given to the CMAC. In this interpretation for an N -variate CMAC every bit in the association vector corresponds to a binary basis function with a compact N -dimensional hypercube support. The size of the hypercube is C quantization intervals. This means that a bit of the association vector will be active if and only if the input vector is within the support of the corresponding basis function. This support is often called receptive field of the basis function [1].

The two mappings are implemented in a two-layer network architecture. The first layer, the association layer, implements a special encoding of the quantized input data. This layer is fixed. The second layer is a linear combiner with the trainable weight vector \mathbf{w} .

2.1 The Training of the CMAC

The weights of a CMAC are determined using a training data set $\{\mathbf{x}(k), y_d(k)\}_{k=1}^P$ where $\mathbf{x}(k) \in Z^N$ is the discrete input and $y_d(k) \in R$ is the corresponding desired output of the k -th training sample. The weights can be determined analytically using matrix inversion, or can be trained using an iterative training algorithm.

Every element of the training data set defines a linear equation for the weight vector

$$y_d(k) = \mathbf{a}(k)^T \mathbf{w} + e(k), \quad k = 1, \dots, P \quad (2)$$

where $\mathbf{a}(k)$ is the association vector for the k -th training sample and $e(k)$ is the error at the output of the network for the same training sample. If all training samples are considered, a linear equation set is obtained, which can be written in the form $\mathbf{y}_d = \mathbf{Aw} + \mathbf{e}$. Here \mathbf{A} is the association matrix formed from the association vectors as row vectors, $\mathbf{y}_d = [y_d(1) \ y_d(2) \ \dots \ y_d(P)]^T$ is the output vector of the desired values of all training data, and \mathbf{e} is the error vector formed from the error values $e(k) = y_d(k) - y(k) = y_d(k) - \mathbf{w}^T \mathbf{a}(k)$, $k = 1, \dots, P$.

The goal of training is to find a weight vector \mathbf{w}^* that minimizes the mean squared error for all training data:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \frac{1}{2} (\mathbf{e}^T \mathbf{e}) = \arg \min_{\mathbf{w}} \frac{1}{2} (\mathbf{y}_d - \mathbf{Aw})^T (\mathbf{y}_d - \mathbf{Aw}) \quad (3)$$

The analytical solution gives the optimal weight vector in the sense of least squares error as:

$$\mathbf{w}^* = \mathbf{A}^\dagger \mathbf{y}_d \quad (4)$$

where $\mathbf{A}^\dagger = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T = \mathbf{A}^T (\mathbf{A} \mathbf{A}^T)^{-1}$ is the pseudo inverse of the association matrix \mathbf{A} .

Although the optimal (in the mean squared error sense) weight vector of a CMAC can be determined analytically usually iterative training is used to find the optimal weight values. This iterative learning rule is usually a version of the LMS algorithm.

An advantage of the CMAC networks is that in one training step only a small fraction of all weights - the C weights selected by the active bits of the association vector - are modified. Moreover these weights are modified in the same way, as for all selected weights Δw_i for all $i : a_i = 1$ are equal. As only a few of all weights are adjusted in one training step, the training of the CMAC is *local* and *incremental*. Incremental training means that training of new samples does not decay the "knowledge" gained from previously trained samples, if the new and the previously trained samples are in sufficiently large distance from each other. Using the simple LMS rule with proper learning rate rather high speed training can be achieved.

As it can be shown using the standard least squared error criterion and a version of the iterative training algorithm the generalization error, may be rather

large [13]. This error can be reduced if a modified training rule is used(Eq. (6)), which can be achieved by using the cost function below:

$$C(k) = \frac{1}{2} (y_d(k) - y(k))^2 + \frac{\lambda}{2} \left(\frac{y_d(k)}{C} - w_i(k) \right)^2 \quad (5)$$

$$w_i(k+1) = w_i(k) + \mu e(k) + \lambda \left(\frac{y_d(k)}{C} - w_i(k) \right) \quad (6)$$

for all weight values with index $i : a_i(k) = 1$. Here the first part is the standard LMS rule, and the second part comes from the special smoothing regularization term (the second term of (5)). Here λ is a regularization parameter, and the values of the learning rate μ and the regularization coefficient λ are responsible for finding a good compromise between the two terms.

2.2 The Complexity of the Network

The way of encoding - the positions of the basis functions in the first layer - and the value of C determine the complexity, and the modeling and generalization properties of the network. Network complexity can be characterized by the number of basis functions what is the same as the size of the weight memory. In one-dimensional case with r discrete input values a CMAC needs $M = r + C - 1$ weight values [12]. However, if we follow this rule in multivariate cases, the number of basis functions and the size of the weight memory will grow exponentially with the input dimension. If there are r_i discrete values for the i -th input dimension an N -dimensional CMAC needs $M = \prod_{i=1}^N (r_i + C - 1)$ weight values.

In multivariate cases the weight memory can be so huge that practically it cannot be implemented. As an example consider a ten-dimensional binary CMAC where all input components are quantized into 10 bits. In this case the size of the weight memory would be enormous, approximately 2^{100} .

To avoid this high complexity the number of basis functions must be reduced. In a classical multivariate CMAC this reduction is achieved by using basis functions positioned only at the diagonals of the quantized input space as it is shown in Fig. 1.

The shaded regions in the figure are the receptive fields of the different basis functions. As it is shown these receptive fields are grouped into overlays. One overlay contains basis functions with non-overlapping supports, but the union of the supports covers the whole input space, that is each overlay covers the whole input N -dimensional lattice. The different overlays have the same structure; they consist of similar basis functions in shifted positions. Every input data will select C basis functions, each of them on a different overlay, so in an overlay one and only one basis function will be active for every input point. The positions of the overlays and the basis functions of one overlay are represented by definite points. In the original Albus scheme the overlay-representing points are in the main diagonal of the input space, whereas the basis-function-positions are represented by the sub-diagonal points, as it is shown in Fig. 1 (black dots).

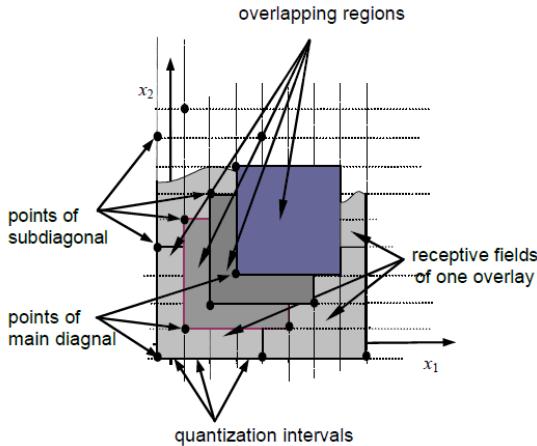


Fig. 1. The receptive fields of a two-variable Albus CMAC

In the original Albus architecture [1] the number of overlays does not depend on the dimension of the input vectors; it is always C . This is in contrast to the so-called full overlay version when C^N overlays are used. In a multivariate Albus CMAC the number of basis function will not grow exponentially with the

$$\text{input dimension, it will be } M = \sum_{i=0}^{C-1} \prod_{j=1}^N \left\lceil \frac{(i+R_j)}{C} \right\rceil \approx \left\lceil \frac{1}{C^{N-1}} \prod_{j=1}^N (R_j + C - 1) \right\rceil,$$

where $\lceil x \rceil$ denotes the function that rounds x toward plus infinity. Using this architecture there will be "only" $\approx 2^{55}$ weight values in the previous example. Although this is a great reduction, the size of the memory is still prohibitively large so further complexity reduction is required. As such large memories cannot be implemented in practice, it is called conceptual memory. For getting an implementable network complexity reduction is required that will result in a much smaller physical memory. In the Albus binary CMAC hash-coding is applied for this purpose. It significantly reduces the size of the (physical) weight memory however it can result in collisions of the mapped weights and some unfavorable effects on the convergence of CMAC learning [8], [9]. Collision means that different addresses of the conceptual memory are mapped into the same physical memory address. The effect is that distant inputs will address partly common weights, perturbing the local generalization capability and causing some interference between distant data - this is caused by the first type of collision -, or that certain outputs are formed as a sum of less than C weights - this is caused by the second type of collision.

3 Kernel CMAC

An entirely different approach to reduce memory complexity is if the kernel representation of the network is applied. Kernel machines [14] have been developed

in the last decade and proved to be efficient new approaches for solving the learning problem from samples.

Kernel machines solve the dimensionality-problem by applying a trick. They also use nonlinear mapping from the input space into the feature space, however, they do not look for the solution in the feature space, instead, the solution is obtained in a new space called kernel space. Kernel space representation is defined easily from the feature space representation using scalar products. The significance of using the kernel trick is that the complexity of the solution is greatly reduced: the dimension of the kernel space and so the size of the weight memory is upper bounded by the number of training samples independently of the dimension of the feature space.

The input-output mapping of a basis function network based on the feature-space representation is:

$$y(\mathbf{x}) = \sum_{j=1}^M w_j \varphi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \quad (7)$$

where $\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \dots, \varphi_M(\mathbf{x})]^T$ is a projection of the input vector \mathbf{x} into the feature space and \mathbf{w} is the weight vector. The same mapping can be described in the kernel space as:

$$y(\mathbf{x}) = \sum_{k=1}^P \alpha_k K(\mathbf{x}, \mathbf{x}(k)) \quad (8)$$

It is also a weighted sum of nonlinear functions where the $K(\mathbf{x}, \mathbf{x}(k)) = \boldsymbol{\varphi}^T(\mathbf{x}) \boldsymbol{\varphi}(\mathbf{x}(k))$, $k = 1, \dots, P$ functions are called kernel functions defined by scalar products, and the α_k coefficients serve as the weight values.

Being a basis function network, CMAC can also be represented as a kernel machine [12]. Here the kernel functions are defined as

$$K(\mathbf{x}, \mathbf{x}(k)) = \mathbf{a}^T(\mathbf{x}) \mathbf{a}(\mathbf{x}(k)) \quad (9)$$

where $\mathbf{a}(\mathbf{x})$ is the association vector for the discrete input \mathbf{x} and the response of a CMAC for a given \mathbf{x} in kernel representation can be written as:

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{a}^T(\mathbf{x}) \mathbf{w}^* = \mathbf{a}^T(\mathbf{x}) \mathbf{A}^T (\mathbf{A} \mathbf{A}^T)^{-1} \mathbf{y}_d \\ &= \mathbf{a}^T(\mathbf{x}) \mathbf{A}^T \boldsymbol{\alpha} = \mathbf{k}^T(\mathbf{x}) \boldsymbol{\alpha} \end{aligned} \quad (10)$$

where

$$\boldsymbol{\alpha} = (\mathbf{A} \mathbf{A}^T)^{-1} \mathbf{y}_d \quad (11)$$

In kernel representation the components of $\boldsymbol{\alpha}$ are considered as the weight values, so instead of using M weights, here only P weights will be used. As in multidimensional cases $P \ll M$ this is a great reduction of memory complexity.

In kernel representation for every training point a kernel function is used positioned in the input space at the given training input point. The support of

a kernel function will be exactly the same as the local generalization area of the corresponding training data. It should be noted that the kernel representation is nothing more than an alternative form of the original CMAC. Its modeling capability is equivalent to the capability of the standard CMAC. The kernel representation has a further advantage. Its complexity does not depend on the dimension of the feature space, the number of kernel functions equals to the number of training points. Kernel representation may help to reduce memory complexity not only for standard CMACs, but for the full overlay version too, where extremely large number of basis functions should be used. Full overlay version is used to increase the modelling capability of a CMAC. In this case the number of overlays is increased up to C^N . The direct implementation of this CMAC version is almost impossible except for very low-dimensional inputs. The kernel representation, however, can be implemented for high-dimensional inputs too. The multivariate kernel functions for a full-overlay binary CMAC can be obtained as tensor product of univariate second order *B-spline functions*.

3.1 Kernel CMAC with Weight-Smoothing Regularization

To improve generalization capability weight smoothing regularization should also be applied for the kernel version [12]. In this case Eq. (10) will be modified

$$y(\mathbf{x}) = \mathbf{a}^T(\mathbf{x})(\mathbf{I} + \lambda \mathbf{D})^{-1} \mathbf{A}^T \left[\boldsymbol{\alpha} + \frac{\lambda}{C} \mathbf{y}_d \right]. \quad (12)$$

Here

$$\boldsymbol{\alpha} = \left(\mathbf{K}_D + \frac{1}{\gamma} \mathbf{I} \right)^{-1} \left(\mathbf{I} - \frac{\lambda}{C} \mathbf{K}_D \right) \mathbf{y}_d \quad (13)$$

where $\mathbf{K}_D = \mathbf{A}(\mathbf{I} + \lambda \mathbf{D})^{-1} \mathbf{A}^T$ and $\mathbf{D} = \sum_{k=1}^P \text{diag}(\mathbf{a}(k))$.

In kernel representation both analytic and iterative solutions can be obtained [12]. To get analytic solution matrix inversion is required where the matrix to be inverted is of dimension $P \times P$. This is the consequence of the kernel representation. In the regularized version $(\mathbf{I} + \lambda \mathbf{D})$ should also be inverted. Although it is a diagonal matrix where inversion is nothing more than reciprocate the diagonal elements, its size is $M \times M$, and M is equal to the length of the association vector and as it was shown before $M \gg P$, that is M may be extremely large. More efficient solution can be obtained if the size of this matrix is reduced.

3.2 Regularized Kernel CMAC with Hash-Coding

The primary goal of hash-coding was to reduce the length of the association vector \mathbf{a} , mapping it into a compressed one \mathbf{a}_c [1]. The drawback of hash-coding is that - because of collisions in the compressed association vector - it deteriorates the performance of the problem. To mitigate or eliminate of this effect two possibilities could be used. Either a properly constructed hash-coding should

be used where the probability of collisions is forced to be small, or smoothing regularization should be applied.

The new proposal applies both of them. A property of the proposed hash-coding is to reach, that the memory addresses close in the conceptual memory should be close in the physical memory. This is good because we assume that the addresses close to each other in the conceptual memory represent basis functions close to each other in the input space. So if we can guarantee that close memory cells remain close after hash-coding, the local generalization capability of the CMAC network remains. This keeps off the first type of collisions of the hash-functions, assuring that distinct inputs will not interfere. The used hash-coding is as follows:

$$H(k) = \left\lceil \frac{k}{M} m_c \right\rceil \quad (14)$$

Here M means the number of basis functions required for the classical CMAC, m_c means the size of the physical memory, and k is the index of the cell in the conceptual memory. So the hash-function just scales down the indexes to the size of the physical memory, but it does not change the order of the cells and it does not make any randomization to the indexes of the cells. However, if we use this type of coding, the second type of collision will occur quite often. So to reach good performance we should allow the network to activate a cell in the physical memory more than once. By doing this we make an implicit regularization, because by selecting the same cell more than once, we achieve a similar effect as with weight smoothing regularization. The result is that the hash-coded association vector will be no longer binary. This is a similar effect as if we had used higher-order CMAC where instead of a binary basis function higher-order B-spline basis functions are used [15]. This rather simple hash-coding works in all such cases well if the number of active bits - which is C in the Albus CMAC and C^N in the full overlay version - is significantly larger than M/m_c .

Using the newly proposed version the basic equations of the regularized kernel CMAC (Eqs (12) and (13)) will be modified as it follows:

$$y(\mathbf{x}) = \mathbf{a}_c^T(\mathbf{x})(\mathbf{I} + \lambda \mathbf{D}_c)^{-1} \mathbf{A}_c^T \left[\boldsymbol{\alpha}_c + \frac{\lambda}{C} \mathbf{y}_d \right] \quad (15)$$

Here

$$\boldsymbol{\alpha}_c = \left(\mathbf{K}_{\mathbf{D}_c} + \frac{1}{\gamma} \mathbf{I} \right)^{-1} \left(\mathbf{I} - \frac{\lambda}{C} \mathbf{K}_{\mathbf{D}_c} \right) \mathbf{y}_d \quad (16)$$

where $\mathbf{K}_{\mathbf{D}_c} = \mathbf{A}_c (\mathbf{I} + \lambda \mathbf{D}_c)^{-1} \mathbf{A}_c^T$ and $\mathbf{D}_c = \sum_{k=1}^P \text{diag}(\mathbf{a}_c(k))$. In these expressions c subscript means that all vectors and matrices are formed after hash-coding. In this version although the length of $\boldsymbol{\alpha}_c$ equals with the length of $\boldsymbol{\alpha}$, as it is determined by the number of training samples, the \mathbf{D}_c will be a matrix of dimension $m_c \times m_c$ only.

4 Illustrative Experimental Results

The proposed versions of the CMAC network were extensively validated experimentally. Here only the results for the simple sinc function approximation will be presented. Fig. 2. shows the responses with $C = 36$ and with the distance between the neighbouring sampling points 20 measured in input quantization units. In Fig. 2a) the result of a classical Albus CMAC can be seen. Fig. 2b) shows when hash-coding is applied without regularization and where the hash-coding defined in [9] is applied. In Fig. 2c) the result of the proposed version is presented. The length of the association vector (Fig. 2a)) is 395, while after hash-coding the compressed association vector is 107 (in case Fig. 2b) and 25 in cases Fig. 2c) and 2d). The difference between Fig. 2c) and 2d) shows the effect of regularization. It can also be seen that even the compressed association vector is much smaller than in case of Fig. 2b) the result is better.

A 2D sinc function is approximated by a kernel CMAC (Fig. 3a), and by a regularized kernel CMAC. The proposed new hash-coding is applied in both cases. The length of the original association vector is 12769, while after hash-coding only 800-bit long compressed association vectors are used.

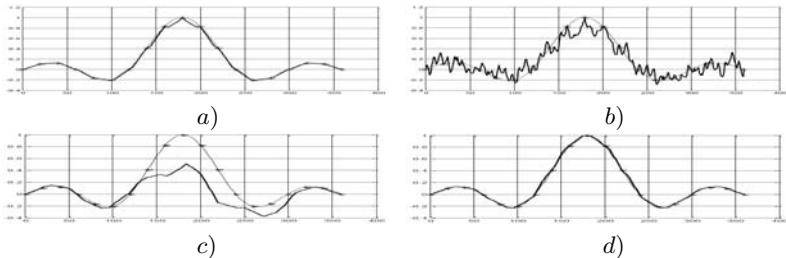


Fig. 2. The response of the CMAC. a.) Albus CMAC, b.) Albus CMAC with hash-coding [9], c.) kernel CMAC with the proposed hash-coding without regularization d.) regularized kernel CMAC with the proposed hash-coding ($\lambda=100$, $\gamma=100$).

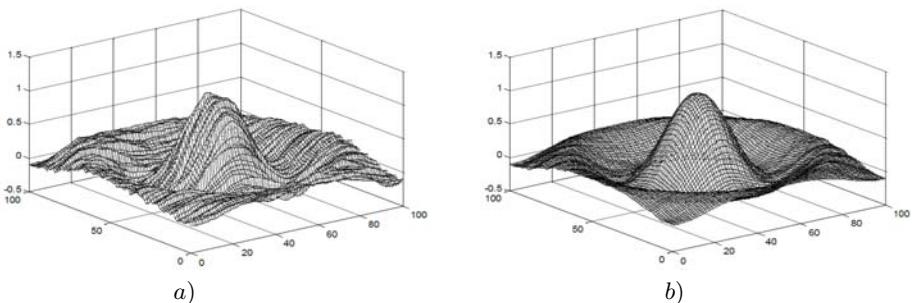


Fig. 3. Response of the hash-coded kernel CMAC a.) without regularization, b.) with regularization ($\lambda=100$, $\gamma=100$)

5 Conclusions

In this paper a new approach is proposed to reduce memory complexity of the CMAC networks without deteriorating its performance. To achieve this goal a rather simple new hash-coding is proposed which is combined with weight smooting regularization. The proposed approach is especially suitable for full overlay kernel representation, where the complexity may be reduced by many order of magnitude, however it can be applied for the original C-overlay version too. In kernel representation memory complexity can further be reduced finding a sparse solution using e.g. kernel pruning [13] or LS2-SVM approach [14].

References

1. Albus, J.S.: A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC). Transaction of the ASME, 220–227 (September 1975)
2. Thompson, D.E., Kwon, S.: Neighbourhood Sequential and Random Training Techniques for CMAC. IEEE Trans. on Neural Networks 6, 196–202 (1995)
3. Ker, J.S., Kuo, Y.H., Wen, R.C., Liu, B.D.: Hardware Implementation of CMAC Neural Network with Reduced Storage Requirement. IEEE Trans. on Neural Networks 8, 1545–1556 (1997)
4. Horváth, G., Deák, F.: Hardware Implementation of Neural Networks Using FPGA Elements. In: Proc. of The International Conference on Signal Processing Application and Technology, Santa Clara, vol. II, pp. 60–65 (1993)
5. Brown, M., Harris, C.J., Parks, P.C.: The Interpolation Capabilities of the Binary CMAC. Neural Networks 6(3), 429–440 (1993)
6. Szabó, T., Horváth, G.: Improving the Generalization Capability of the Binary CMAC. In: Proc. Int. Joint Conf. on Neural Networks, IJCNN 2000, Como, Italy, vol. 3, pp. 85–90 (2000)
7. Zhong, L., Zhongming, Z., Chongguang, Z.: The Unfavorable Effects of Hash Coding on CMAC Convergence and Compensatory Measure. In: IEEE International Conference on Intelligent Processing Systems, Beijing, China, pp. 419–422 (1997)
8. Wang, Z.Q., Schiano, J.L., Ginsberg, M.: Hash Coding in CMAC Neural Networks. In: Proc. of the IEEE International Conference on Neural Network, Washington, USA, vol. 3, pp. 1698–1703 (1996)
9. Li, C.K., Chiang, C.T.: Neural Networks Composed of Single-variable CMACs. In: Proc. of the 2004 IEEE International Conference on Systems, Man and Cybernetics, The Hague, The Netherlands, pp. 3482–3487 (2004)
10. Lee, H.M., Chen, C.M., Lu, Y.F.: A Self-Organizing HCMAC Neural-Network Classifier. IEEE Trans. on Neural Networks 14, 15–27 (2003)
11. Hung, S.L., Jan, J.C.: MS_CMAC Neural Network Learning Model in Structural Engineering. Journal of Computing in Civil Engineering, 1–11 (January 1999)
12. Horváth, G., Szabó, T.: Kernel CMAC with improved capability. IEEE Trans. Sys. Man Cybernet. B 37(1), 124–138 (2007)
13. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific, Singapore (2002)
14. Valyon, J., Horváth, G.: Selection methods for extended least squares support vector machines. Int. Journal of Intelligent Computing and Cybernetics 1(1), 69–93 (2008)
15. Lane, S.H., Handelman, D.A., Gelfand, J.J.: Theory and Development of Higher-Order CMAC Neural Networks. IEEE Control Systems Magazine 12(2), 23–30 (1992)

Model Complexity of Neural Networks and Integral Transforms

Věra Kůrková

Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague

vera@cs.cas.cz

<http://www.cs.cas.cz/~vera>

Abstract. Model complexity of neural networks is investigated using tools from nonlinear approximation and integration theory. Estimates of network complexity are obtained from inspection of upper bounds on decrease of approximation errors in approximation of multivariable functions by networks with increasing numbers of units. The upper bounds are derived using integral transforms with kernels corresponding to various types of computational units. The results are applied to perceptron networks.

1 Introduction

Integral transformations (such as Fourier, Gabor or Radon) play an important role in applied science. Also functions computable by neural-network units can be used as kernels of integral transformations. First applications of integral transforms in mathematical theory of neurocomputing occurred in proofs of the universal approximation property. Carroll and Dickinson [1] and Ito [2] used Radon transform to show that all smooth continuous functions can be represented as integrals in the form of networks with infinitely many perceptrons. They derived universal approximation property of perceptron networks from discretization of such representations. Park and Sandberg [3,4] employed representations of L^p -functions as limits of convolutions with properly scaled kernel or radial functions to prove the universal approximation property of radial-basis function networks.

Integral transforms in the form of networks with infinitely many units were also used to obtain estimates of network complexity. Such estimates can be derived from inspection of upper bounds on speed of decrease of errors in approximation of multivariable functions by networks with an increasing number of units. Jones [5, p.612] proved such an upper bound and suggested applying this bound to functions with representations as infinite networks with trigonometric perceptrons. Barron [6] refined Jones' result and proved an estimate of model complexity for sigmoidal perceptron networks using weighted Fourier transform. Girosi and Anselotti [7] applied the estimate by Jones and Barron, known as Maurey-Jones-Barron's theorem, to convolutions with suitable kernels. Kůrková et al. [8] and Kainen et al. [9] applied this estimate to representations in the form of networks with infinitely many Heaviside perceptrons.

In this paper, we present a unifying framework for estimation of model complexity for networks with general types of computational units based on integral transforms induced by such units. We use a reformulation of Maurey-Jones-Barron's theorem and its extensions in terms of certain variational norms tailored to network units. We derive an estimate of these norms for functions representable as integrals in the form of infinite neural networks. Various special cases of this estimate were proven earlier using a variety of proof techniques (such as probabilistic argument [6], approximation of integrals by Riemann sums [8], and Bochner integral [7,10]). However, our results hold under minimal assumptions on a type of hidden units and an ambient function space and their proofs are quite simple thanks to a geometric characterization of the variational norm. Moreover, our estimates allow applications to classes of networks to which previous results have not been applicable.

The paper is organized as follows. In section 2, we introduce notation and describe integral operators defined by computational units. In section 3, we state estimates of rates of approximation over neural networks in terms of variational norms. In section 4, we present our main results on geometric characterization of variational norm, from which in section 5 we derive estimates of variation of functions obtained by integral transforms with kernels corresponding to general computational units. In section 6, we illustrate our results by an example of perceptron networks.

2 Integral Transforms Induced by Computational Units

Neural-network units compute functions depending on parameters (such as weights, biases, centroids). So formally they can be described as mappings

$$\phi : \Omega \times A \rightarrow \mathbb{R},$$

where Ω is a set of variables and A is a set of parameters. Usually, $\Omega \subseteq \mathbb{R}^d$ and $A \subseteq \mathbb{R}^s$. We denote by

$$G_\phi = G_\phi(A) = \{\phi(., a) \mid a \in A\}$$

the parameterized set of functions determined by ϕ .

For example, *perceptrons with an activation function* $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ can be described by a mapping ϕ_σ on \mathbb{R}^{d+1} defined for $(v, b) \in \mathbb{R}^d \times \mathbb{R} = \mathbb{R}^{d+1}$ as

$$\phi_\sigma(x, v, b) = \sigma(v \cdot x + b). \quad (1)$$

Similarly, *kernel units* with a kernel function $\beta : \mathbb{R}^d \rightarrow \mathbb{R}$ can be described by a mapping ϕ_β on \mathbb{R}^{d+1} defined for $(v, b) \in \mathbb{R}^d \times \mathbb{R} = \mathbb{R}^{d+1}$ as

$$\phi_\beta(x, v, b) = \beta(b(x - v)). \quad (2)$$

A special case are *radial units* with $\beta(x) = \alpha(\|x\|)$, where $\alpha : \mathbb{R}_+ \rightarrow \mathbb{R}$.

The set of input-output functions of a network with one-hidden-layer of n units computing ϕ is of the form

$$\text{span}_n G_\phi(A) = \left\{ \sum_{i=1}^n w_i \phi(., a_i) \mid w_i \in \mathbb{R}, a_i \in A \right\}.$$

A network unit computing a function $\phi : \Omega \times A \rightarrow \mathbb{R}$ induces an integral operator. Such an operator depends on a measure μ on A and is defined for a function w in a suitable function space as

$$L_\phi(w)(x) = \int_A w(a) \phi(x, a) d\mu(a). \quad (3)$$

Metaphorically, the integral on the right-hand side of the equation (3) can be interpreted as a *one-hidden-layer neural network with infinitely many units* computing functions $\phi(., a)$ with $a \in A$. So L_ϕ is an operator transforming output-weight functions w of infinite networks to their input-output functions $L_\phi(w)$.

The classes of functions which can be represented as integrals in the form of infinite neural networks with various types of units are quite large. For example, all continuous compactly supported functions and all L^p -functions with $p \in [1, \infty)$ can be expressed as limits of sequences of input-output functions of infinite networks with quite general radial or kernel functions [34]. Also all functions from Sobolev spaces (i.e., functions satisfying certain conditions on their derivatives) can be represented as infinite networks with Gaussian radial functions [1112] and sufficiently smooth functions decreasing rapidly at infinity (in particular, the Gaussian function) can be expressed as networks with infinitely many sigmoidal perceptrons [2813].

3 Rates of Approximation by Neural Networks

An importance of the role of integral transforms induced by computational units in investigation of model complexity of neural networks follows from the role of such transforms in estimation of certain norms tailored to computational units. Values of these norms are critical factors in upper bounds on rates of approximation by neural networks in functions spaces of various types.

Such norms are defined quite generally. For G a bounded nonempty subset of a normed linear space $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, the norm G -variation, denoted $\|\cdot\|_G$, is defined for all $f \in \mathcal{X}$ as

$$\|f\|_G = \|f\|_{G, \mathcal{X}} = \inf \left\{ c > 0 \mid c^{-1} f \in \text{cl conv}(G \cup -G) \right\},$$

where the closure cl is taken with respect to the topology generated by the norm $\|\cdot\|_{\mathcal{X}}$ and conv denotes the convex hull. So G -variation depends on the ambient space norm, but when it is clear from the context, we write merely $\|f\|_G$ instead of $\|f\|_{G, \mathcal{X}}$. It is easy to check that G -variation is a norm on the subspace of \mathcal{X} formed by those f for which $\|f\|_G$ is finite (G -variation is the Minkowski functional of the closed convex symmetric hull of G).

The following theorem gives upper bounds on rates of approximation by sets of the form $\text{span}_n G$. It is a reformulation of results by Maurey [14], Jones [5], Barron [6], and Darken et. al. [15] in terms of G -variation (see [16][17]).

Theorem 1. *Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be a normed linear space, G its bounded nonempty subset, $s_G = \sup_{g \in G} \|g\|_{\mathcal{X}}$, $f \in \mathcal{X}$ and n be a positive integer. Then*

(i) for $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ a Hilbert space,

$$\|f - \text{span}_n G\|_{\mathcal{X}}^2 \leq \frac{s_G^2 \|f\|_G^2 - \|f\|_{\mathcal{X}}^2}{n};$$

(ii) for $(\mathcal{X}, \|\cdot\|_{\mathcal{X}}) = (\mathcal{L}^p(\Omega, \rho), \|\cdot\|_{\mathcal{L}^p})$, $p \in (1, \infty)$, and ρ a measure on $\Omega \subseteq \mathbb{R}^d$,

$$\|f - \text{span}_n G\|_{\mathcal{L}^p} \leq \frac{2^{1+1/r} s_G \|f\|_G}{n^{1/s}},$$

where $1/q + 1/p = 1$, $r = \min(p, q)$, $s = \max(p, q)$.

Theorem 1 implies estimates of model complexity needed to guarantee a given accuracy of approximation for all functions from balls of a given radii in variational norms. By (i), any number n of units satisfying

$$n \geq \frac{s_G^2 r^2}{\varepsilon^2}$$

is sufficient to guarantee approximation within ε for all functions from the ball of radius r in G -variation.

4 Properties of Variational Norms

To apply Theorem 1 to neural networks, one needs to estimate $G_\phi(A)$ -variation for various types of hidden-unit functions ϕ . In this section, we derive properties of variational norms which enable us to estimate $G_\phi(A)$ -variation for functions representable as integrals in the form of infinite networks. We first prove a characterization of variational norm in terms of bounded linear functionals. Although it is rather abstract, this characterization leads to a simple proof of an upper bound on G -variation in terms of the \mathcal{L}^1 -norm of the output-weight function.

By \mathcal{X}^* is denoted the *dual* of \mathcal{X} (the space of all bounded linear functionals on \mathcal{X}) and by G^\perp the *orthogonal complement* of a subset G of \mathcal{X} , i.e., the set of all elements l of \mathcal{X}^* such that for some $g \in G$, $l(g) \neq 0$.

Theorem 2. *Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be a normed linear space, G be its nonempty bounded subset and $f \in \mathcal{X}$ be such that $\|f\|_G < \infty$. Then*

$$\|f\|_G = \sup_{l \in \mathcal{X}^* \setminus G^\perp} \frac{|l(f)|}{\sup_{g \in G} |l(g)|}.$$

Proof. First, we show that for all $c > 0$ and $f \in \mathcal{X}$

$$f/c \in \text{cl conv}(G \cup -G) \iff \forall l \in \mathcal{X}^* \quad |l(f)| \leq c \sup_{g \in G} |l(g)|. \quad (4)$$

Assume that $f/c \in \text{cl conv}(G \cup -G)$. Then there exists a sequence $\{f_k\}$ such that $\lim_{k \rightarrow \infty} \|f/c - f_k\|_{\mathcal{X}} = 0$ and all f_k can be represented as $f_k = \sum_{i=1}^{m_k} w_{k,i} g_{k,i}$, where $\sum_{i=1}^{m_k} |w_{k,i}| \leq 1$ and all $g_{k,i} \in G$. Then for all $l \in \mathcal{X}^*$, $l(f_k) = \sum_{i=1}^{m_k} w_{k,i} l(g_{k,i})$ and so $|l(f_k)| \leq \sup_{g \in G} |l(g)|$. Since l is continuous, also $|l(f/c)| \leq \sup_{g \in G} |l(g)|$ and thus $|l(f)| \leq c \sup_{g \in G} |l(g)|$.

If $f/c \notin \text{cl conv}(G \cup -G)$, then by Mazur's theorem [18, p.108] there exists $l \in \mathcal{X}^*$ such that $|l(f/c)| > 1$ and for all $h \in \text{cl conv}(G \cup -G)$, $|l(h)| \leq 1$. Thus in particular for all $g \in G$, $|l(g)| \leq 1$. Hence $|l(f)| = l(f) > c \geq c \sup_{g \in G} |l(g)|$ and so (4) holds.

By (4), for all c for which $f/c \in \text{cl conv}(G \cup -G)$, $\sup_{l \in \mathcal{X}^* \setminus G^\perp} \frac{|l(f)|}{\sup_{g \in G} |l(g)|} \leq c$. Thus $\|f\|_G = \inf\{c > 0 \mid f/c \in \text{cl conv}(G \cup -G)\} \geq \sup_{l \in \mathcal{X}^* \setminus G^\perp} \frac{|l(f)|}{\sup_{g \in G} |l(g)|}$.

To prove the opposite inequality $\|f\|_G \leq \sup_{l \in \mathcal{X}^* \setminus G^\perp} \frac{|l(f)|}{\sup_{g \in G} |l(g)|}$ by contradiction, assume that $\|f\|_G > b = \sup_{l \in \mathcal{X}^* \setminus G^\perp} \frac{|l(f)|}{\sup_{g \in G} |l(g)|}$. Then by (4), there exists $l_b \in \mathcal{X}^*$ such that $|l_b(f)| > b \sup_{g \in G} |l_b(g)|$. By the definition of b , this can only hold when $l_b \in G^\perp$. But as we assume that $\|f\|_G$ is finite, there exists some $c > 0$ for which $f/c \in \text{cl conv}(G \cup -G)$. By (4), $|l_b(f)| \leq c \sup_{g \in G} |l_b(g)|$. This implies that $l_b(f) = 0$ because $l_b \in G^\perp$. Thus we get a contradiction with the inequality $|l_b(f)| > b \sup_{g \in G} |l_b(g)|$. \square

Theorem 2 is an extension of a geometric characterization of variational norm in Hilbert spaces proven in [19]. As all bounded linear functionals on a Hilbert space are inner products [20, p. 206], in the case of \mathcal{X} a Hilbert space, Theorem 2 states

$$\|f\|_G = \sup_{h \in \mathcal{X} \setminus G^\perp} \frac{|f \cdot h|}{\sup_{g \in G} |g \cdot h|}.$$

This implies,

$$\|f\|_G \geq \frac{\|f\|^2}{\sup_{g \in G} |f \cdot g|},$$

which means that G -variation is large for functions which are almost orthogonal to the set G .

5 Estimates of Variation of Functions Representable as Integral Transforms

In this section, we prove our main result giving an upper bound on variation of functions representable as integrals in the form of networks with infinitely many units.

First, consider the case when the set A of parameters is finite. Each ordering $A = \{a_1, \dots, a_m\}$ determines a linear operator $L_\phi : \mathbb{R}^m \rightarrow \mathcal{X}$ defined as $L_\phi(w)(x) = \sum_{i=1}^m w_i \phi(x, a_i)$, where $w = (w_1, \dots, w_m)$. It follows easily from the definition of variation that for each f which can be represented as $f = L_\phi(w)$,

$$\|f\|_{G_\phi(A)} \leq \|w\|_1 = \sum_{i=1}^m |w_i|. \quad (5)$$

Note that although f can be represented as an input-output function of a finite neural network, this network might be quite large. In such a case, estimates of rates of approximation of f by input-output functions of smaller networks obtained from Theorem 1 might be useful.

When the set A of parameters is infinite, analogy with (5) suggests that for f representable as $f(x) = \int_A w(a) \phi(x, a) d\mu(a)$, the estimate

$$\|f\|_{G_\phi(A)} \leq \|w\|_{\mathcal{L}^1(A, \mu)} \quad (6)$$

might hold. The inequality (6) can only be considered when quantities on both its sides are well defined, i.e., when

- (i) $G_\phi(A)$ is a bounded subset of $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ and
- (ii) $w \in \mathcal{L}^1(A, \mu)$.

Our main result (Theorem 3) shows that in a wide class of function spaces, the assumptions (i) and (ii) are sufficient to guarantee the relationship (6) between $G_\phi(A)$ -variation and \mathcal{L}^1 -norm.

To prove the theorem, we first describe classes of function spaces with a certain commutativity property of bounded linear functionals. In spaces with this property, the inequality (6) can be easily derived using Theorem 2. We denote by $(\mathcal{C}_c(\Omega), \|\cdot\|_{\sup})$ the space of all continuous compactly supported functions on $\Omega \subseteq \mathbb{R}^d$ with the supremum norm and by $(\mathcal{C}_0(\mathbb{R}^d), \|\cdot\|_{\sup})$ the space of all continuous functions on \mathbb{R}^d that vanish at infinity.

Proposition 1. *Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be one of the following spaces:*

- (i) $(\mathcal{L}^p(\Omega, \rho), \|\cdot\|_{\mathcal{L}^p})$ with $p \in [1, \infty)$, $\Omega \subseteq \mathbb{R}^d$, and ρ a σ -finite measure,
- (ii) $(\mathcal{C}_c(\Omega), \|\cdot\|_{\sup})$ with Ω a locally compact subset of \mathbb{R}^d ,
- (iii) $(\mathcal{C}_0(\Omega), \|\cdot\|_{\sup})$ with $\Omega = \mathbb{R}^d$.

Let $\phi : \Omega \times A \rightarrow \mathbb{R}$ be such that $G_\phi(A) = \{\phi(., a) \mid a \in A\}$ is a bounded subset of $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, $w \in \mathcal{L}^1(A, \mu)$, where μ is a σ -finite measure and $f \in \mathcal{X}$ be such that for all $x \in \Omega$, $f(x) = \int_A w(a) \phi(x, a) d\mu(a)$. Then for all $l \in \mathcal{X}^*$,

$$l(f) = \int_A w(a) l(\phi(., a)) d\mu(a).$$

Proof. First, we prove the statement for the case (i). By the properties of the duals of \mathcal{L}^p -spaces with $p \in [1, \infty)$ [20, pp. 176, 180], for every $l \in \mathcal{X}^*$ there exists $h \in \mathcal{L}^q(\Omega, \rho)$, where for $p > 1$, $1/q + 1/p = 1$ and for $p = 1$, $q = \infty$, such that for

all $f \in \mathcal{L}^p(\Omega, \rho)$, $l(f) = \int_{\Omega} f(x)h(x)d\rho(x)$. By Hölder's inequality [20, p. 96] for all $a \in A$, $\phi(., a)h \in \mathcal{L}^1(\Omega, \rho)$ and $\|\phi(., a)h\|_{\mathcal{L}^1} \leq \|\phi(., a)\|_{\mathcal{L}^p}\|h\|_{\mathcal{L}^q}$. Thus for all $a \in A$, $\int_{\Omega} |\phi(x, a)h(x)|d\rho(x) \leq \|\phi(., a)\|_{\mathcal{L}^p}\|h\|_{\mathcal{L}^q}$. As $\sup_{a \in A} \|\phi(., a)\|_{\mathcal{L}^p} = s_{\phi}$ is finite, $\int_{\Omega} \int_A |w(y)\phi(x, y)h(x)|d\mu(y)d\rho(x) \leq s_{\phi}\|w\|_{\mathcal{L}^1}$ is finite. Thus we can use Fubini's theorem [20, p. 86] to obtain

$$\begin{aligned} l(f) &= \int_{\Omega} \left(\int_A w(a)\phi(x, a)d\mu(a) \right) h(x)d\rho(x) = \\ &\int_A w(a) \left(\int_{\Omega} \phi(x, a)h(x)d\rho(x) \right) d\mu(a) = \int_A w(a)l(\phi(., a))d\mu(a). \end{aligned}$$

The proof for the cases (ii) and (iii) is analogous to the case (i). The only difference is that the characterization of bounded linear functionals on \mathcal{L}^p is replaced with the Riesz Representation Theorem [21]. By this theorem for every $l \in \mathcal{X}^*$, there exists a signed measure ν on Ω such that for all $f \in \mathcal{C}_c(\Omega)$ or $f \in \mathcal{C}_0(\Omega)$, $l(f) = \int_{\Omega} f(x)d\nu(x)$ and $|\nu|(\Omega) = \|l\|$, where $|\nu|$ denotes the total variation of ν . Thus for all $a \in A$, $\int_{\Omega} |\phi(x, a)|d\nu(x) \leq \|\phi(., a)\|_{\sup}|\nu|(\Omega)$. As $\sup_{a \in A} \|\phi(., a)\|_{\sup} = s_{\phi}$ is finite, also $\int_A |w(a)| \int_{\Omega} |\phi(x, a)|d\nu(x)d\mu(a) \leq s_{\phi}\|l\| \int_A |w(a)|d\mu(a) \leq s_{\phi}\|l\|\|w\|_{\mathcal{L}^1}$ is finite. Thus we can use Fubini's theorem to obtain

$$\begin{aligned} l(f) &= \int_{\Omega} \left(\int_A w(a)\phi(x, a)d\mu(a) \right) d\nu(x) = \int_A w(a) \left(\int_{\Omega} \phi(x, a)d\nu(x) \right) d\mu(a) = \\ &\int_A w(a)l(\phi(., a))d\mu(a). \quad \square \end{aligned}$$

Theorem 2 combined with Proposition I gives a simple proof of the upper bound (6) on G_{ϕ} -variation of functions representable as integral transforms with the kernel ϕ .

Theorem 3. Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be one of the following spaces:

- (i) $(\mathcal{L}^p(\Omega, \rho), \|\cdot\|_{\mathcal{L}^p})$ with $q \in [1, \infty)$, $\Omega \subseteq \mathbb{R}^d$, and ρ a σ -finite measure,
- (ii) $(\mathcal{C}_c(\Omega), \|\cdot\|_{\sup})$ with Ω a locally compact subset of \mathbb{R}^d ,
- (iii) $(\mathcal{C}_0(\Omega), \|\cdot\|_{\sup})$ with $\Omega = \mathbb{R}^d$.

Let $\phi : \Omega \times A \rightarrow \mathbb{R}$ be such that $G_{\phi}(A) = \{\phi(., a) \mid a \in A\}$ is a bounded subset of $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, $w \in \mathcal{L}^1(A)$, where μ is a σ -finite measure, and $f \in \mathcal{X}$ be such that for all $x \in \Omega$, $f(x) = \int_A w(a)\phi(x, a)d\mu(a)$. Then

$$\|f\|_{G_{\phi}(A)} \leq \|w\|_{\mathcal{L}^1(A, \mu)}.$$

Proof. By Proposition I for all $l \in \mathcal{X}^*$, $l(f) = \int_A w(a)l(\phi(., a))d\mu(a)$. Thus $|l(f)| \leq \sup_{a \in A} |l(\phi(., a))| \int_A |w(a)|d\mu(a) = \sup_{a \in A} |l(\phi(., a))| \|w\|_{\mathcal{L}^1(A, \mu)}$. So by Theorem 2

$$\|f\|_{G_{\phi}(A)} = \sup_{l \in \mathcal{X}^* - G_{\phi}(A)^{\perp}} \frac{|l(f)|}{\sup_{a \in A} |l(\phi(., a))|} \leq \|w\|_{\mathcal{L}^1(A, \mu)}. \quad \square$$

Various special cases of this theorem have been derived by a variety of proof techniques, but they all required some restrictions on the space \mathcal{X} (e.g., boundedness of evaluation functionals [10]), on Ω and A (e.g., compactness [8]), and on ϕ and w (e.g., continuity [10]). Theorem 3 only assumes that the output-weight function w is in $\mathcal{L}^1(A, \mu)$ (so that the upper bound exists) and that the set $G_\phi(A)$ is bounded (which is necessary for the definition of $G_\phi(A)$ -variation).

Combining Theorem 3 with estimates from Theorem 1 we get an upper bound on decrease of errors in approximation by neural networks with increasing numbers of units.

Corollary 1. *Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be a space of functions on $\Omega \subseteq \mathbb{R}^d$, $A \subseteq \mathbb{R}^s$, μ be a Borel measure on A . Let $\phi : \Omega \times A \rightarrow \mathbb{R}$ be a mapping such that $G_\phi(A) = \{\phi(\cdot, a) | a \in A\}$ is a bounded subset of $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, and $s_\phi = \sup_{a \in A} \|\phi(\cdot, a)\|_{\mathcal{X}}$. Let $f \in (\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be such that for some $w \in \mathcal{L}^1(A, \mu)$, $f(x) = \int_A w(a) \phi(x, a) d\mu(a)$. Then for all n*

(i) *for $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ a Hilbert space,*

$$\|f - \text{span}_n G_\phi(A)\|_{\mathcal{X}}^2 \leq \frac{s_\phi^2 \|w\|_{\mathcal{L}^1(A, \mu)}^2 - \|f\|_{\mathcal{X}}^2}{n};$$

(ii) *for $(\mathcal{X}, \|\cdot\|_{\mathcal{X}}) = (\mathcal{L}^p(\Omega, \rho), \|\cdot\|_{\mathcal{L}^p})$, $p \in (1, \infty)$, and ρ a σ -finite measure on Ω ,*

$$\|f - \text{span}_n G_\phi(A)\|_{\mathcal{L}^p} \leq \frac{2^{1+1/r} s_\phi \|w\|_{\mathcal{L}^1}}{n^{1/s}},$$

where $1/q + 1/p = 1$, $r = \min(p, q)$, and $s = \max(p, q)$.

6 Rates of Approximation by Perceptron Networks

In this section, we apply our results to perceptron networks. Let $\vartheta : \mathbb{R} \rightarrow \mathbb{R}$ denote the *Heaviside activation function* ($\vartheta(t) = 0$ for $t < 0$ and $\vartheta(t) = 1$ for $t \geq 0$). For a subset Ω of \mathbb{R}^d , let $G_{\vartheta_\vartheta}(\Omega) = \{\vartheta(e \cdot \cdot + b) : \Omega \rightarrow \mathbb{R} | e \in S^{d-1}, b \in \mathbb{R}\}$ (where S^{d-1} is the unit sphere in \mathbb{R}^d) denote the set of functions on Ω computable by Heaviside perceptrons.

For the *Lebesgue measure* λ , we write shortly $\mathcal{L}^2(\Omega)$ instead of $\mathcal{L}^2(\Omega, \lambda)$. By $\mathcal{C}_c^\infty(\mathbb{R}^d)$, $\mathcal{C}_c^d(\mathbb{R}^d)$, resp., is denoted the *space of all compactly supported functions on \mathbb{R}^d with continuous derivatives of all orders and all orders up to d* , resp. By $D_e^{(d)}$ is denoted the *directional derivative* of the order d in the direction of the unit d -dimensional vector e and by $H_{e,b}$ the hyperplane $\{x \in \mathbb{R}^d | \vartheta(e \cdot x + b) = 0\}$. The next theorem gives an upper bound on rates of approximation by Heaviside perceptrons in the space $\mathcal{L}^2(\Omega)$.

Theorem 4. *Let $\Omega \subset \mathbb{R}^d$ be compact with d odd. Then for every $f \in \mathcal{C}_c^d(\mathbb{R}^d)$ and for all n ,*

$$\|f|_\Omega - \text{span}_n G_{\vartheta_\vartheta}(\Omega)\|_{\mathcal{L}^2(\Omega)} \leq \frac{\lambda(\Omega) \|w_f\|_{\mathcal{L}^1(S^{d-1} \times \mathbb{R})}}{\sqrt{n}},$$

where $w_f(e, b) = a(d) \int_{H_{e,b}} (D_e^{(d)}(f))(y) dy$ with $a(d) = (-1)^{(d-1)/2} (1/2)(2\pi)^{1-d}$.

Proof. An integral representation in the form

$$f(x) = \int_{S^{d-1} \times \mathbb{R}} w_f(e, b) \vartheta(e \cdot x + b) d\omega db$$

was derived for all functions from $\mathcal{C}_c^\infty(\mathbb{R}^d)$ by Ito [2] using the Radon transform and for all functions from $\mathcal{C}_c^d(\mathbb{R}^d)$ by Kůrková et al. [8] using properties of the Dirac delta function. As $G_{\phi_\theta}(\Omega)$ is a bounded subset of $\mathcal{L}^2(\Omega)$ with $s_{\phi_\theta} \leq \lambda(\Omega)$, w_f is in $\mathcal{L}^1(S^{d-1} \times \mathbb{R})$, and also we have $f|_\Omega \in \mathcal{L}^2(\Omega)$. Thus we can apply Corollary I(i) to obtain the statement. \square

This estimate gives some insight into the impact of dimensionality on the speed of decrease of approximation errors: the parameter $a(d)$ decreases to zero exponentially fast with d increasing and the \mathcal{L}^1 -norm of the output weight function $\|w_f\|_{\mathcal{L}^1(S^{d-1} \times \mathbb{R})} = a(d) \|\int_{H_{e,b}} (D_e^{(d)}(f))(y) dy\|_{\mathcal{L}^1(S^{d-1} \times \mathbb{R})}$ is bounded from above by the product of $a(d)$ and the maximum of the \mathcal{L}^1 -norms of all iterated partial derivatives of the order d of f (see [9]).

Note that $\lambda(\Omega)$ depends on the shape of the d -dimensional domain $\Omega \subset \mathbb{R}^d$. When Ω is the Euclidean d -dimensional ball, then $\lambda(\Omega)$ goes to zero exponentially fast with d increasing, while when Ω is a cube with the side larger than one, $\lambda(\Omega)$ increases exponentially fast.

Here, we have stated the estimate only for d odd, as for d even the output-weight function w_f in the representation of a compactly supported smooth function f as an integral in the form of infinite perceptron network is more complicated (see [2]).

Acknowledgement

This work was partially supported by GA ČR grant 201/08/1744 and the Institutional Research Plan AV0Z10300504.

References

1. Carroll, S.M., Dickinson, B.W.: Construction of neural net using the radon transform. In: Proc. IJCN, vol. I, pp. 607–611 (1989)
2. Ito, Y.: Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory. Neural Networks 4, 385–394 (1991)
3. Park, J., Sandberg, I.: Universal approximation using radial–basis–function networks. Neural Computation 3, 246–257 (1991)
4. Park, J., Sandberg, I.: Approximation and radial basis function networks. Neural Computation 5, 305–316 (1993)
5. Jones, L.K.: A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. Annals of Statistics 20, 608–613 (1992)
6. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory 39, 930–945 (1993)

7. Girosi, F., Anzellotti, G.: Rates of convergence for Radial Basis Functions and neural networks. In: Mammone, R.J. (ed.) *Artificial Neural Networks for Speech and Vision*, pp. 97–113. Chapman & Hall, Boca Raton (1993)
8. Kůrková, V., Kainen, P.C., Kreinovich, V.: Estimates of the number of hidden units and variation with respect to half-spaces. *Neural Networks* 10, 1061–1068 (1997)
9. Kainen, P.C., Kůrková, V., Vogt, A.: A Sobolev-type upper bound for rates of approximation by linear combinations of Heaviside plane waves. *J. of Approximation Theory* 147, 1–10 (2007)
10. Kainen, P.C., Kůrková, V.: An integral upper bound for neural network approximation. *Neural Computation* (to appear, 2009)
11. Girosi, F.: Approximation error bounds that use VC- bounds. In: Proceedings of the International Conference on Artificial Neural Networks, Paris, pp. 295–302 (1995)
12. Kainen, P.C., Kůrková, V., Sanguineti, M.: Complexity of Gaussian radial basis networks approximating smooth functions. *J. of Complexity* 25, 63–74 (2009)
13. Kainen, P.C., Kůrková, V., Vogt, A.: Integral combinations of heavisides. *Mathematische Nachrichten* (to appear, 2009)
14. Pisier, G.: Remarques sur un résultat non publié de B. Maurey. In: Séminaire d'Analyse Fonctionnelle 1980-1981, École Polytechnique, Centre de Mathématiques, Palaiseau, France, vol. I(12) (1981)
15. Darken, C., Donahue, M., Gurvits, L., Sontag, E.: Rate of approximation results motivated by robust neural network learning. In: Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, pp. 303–309. The Association for Computing Machinery, New York (1993)
16. Kůrková, V.: High-dimensional approximation and optimization by neural networks. In: Suykens, J., Horváth, G., Basu, S., Micchelli, C., Vandewalle, J. (eds.) *Advances in Learning Theory: Methods, Models and Applications*, ch. 4, pp. 69–88. IOS Press, Amsterdam (2003)
17. Kůrková, V., Sanguineti, M.: Error estimates for approximate optimization by the extended Ritz method. *SIAM J. on Optimization* 15, 461–487 (2005)
18. Yoshida, K.: *Functional Analysis*. Springer, Berlin (1965)
19. Kůrková, V., Savický, P., Hlaváčková, K.: Representations and rates of approximation of real-valued Boolean functions by neural networks. *Neural Networks* 11, 651–659 (1998)
20. Friedman, A.: *Learning and Soft Computing*. Dover, New York (1982)
21. Rudin, W.: *Real and Complex Analysis*. MacGraw-Hill, New York (1974)

Function Decomposition Network

Yevgeniy Bodyanskiy, Sergiy Popov, and Mykola Titov

Control Systems Research Laboratory,
Kharkiv National University of Radio Electronics
serge.popov@gmx.net

Abstract. Novel neural network architecture is proposed to solve the nonlinear function decomposition problem. Top-down approach is applied that does not require prior knowledge about the functions properties. Abilities of our method are demonstrated using synthetic test functions and confirmed by a real-world problem solution. Possible directions for further development of the presented approach are discussed.

Keywords: Neural networks, nonlinear function decomposition.

1 Introduction

It is a quite widespread situation in practice, when a variable under investigation y is influenced by many different factors x_1, x_2, \dots, x_n in a complex unknown manner. Understanding the nature of the influence of each factor can aid in many analysis, design and decision making problems and therefore it is often very important for engineers, designers, decision makers, etc. In simple cases, when the process is well-studied, linear or systems first principles are known, the influence of particular factors may be expressed analytically or readily identified from observations by linear modeling, or derived from first principles. However, in more complex systems these approaches become less helpful, since influences become nonlinear and interdependent. In a general case the dependence between x_1, x_2, \dots, x_n and y becomes an arbitrary nonlinear function

$$y = g(x_1, x_2, \dots, x_n) + \xi, \quad (1)$$

where ξ is a noise term with zero mean and bounded variance.

If enough simultaneous observations of x_1, x_2, \dots, x_n and y are available, the problem of estimation of $g(\cdot)$ can be solved by nonlinear identification methods [1], among which artificial neural networks [2], [3] and neuro-fuzzy systems [4], [5], [6] are especially suitable due to their universal approximation capabilities [7]. However, if the systems first principles are not known, i.e. the structure of the model cannot be chosen accordingly, the resulting model is a "black box" and its parameters have no physical sense. Moreover, in a nonlinear case it is hard to perform independent sensitivity analysis for inputs x_i ($i = 1, \dots, n$) because of possible interdependencies between them: the form of influence of a particular x_i may strongly depend on the values of other inputs.

Consider, for example, dependence of the form

$$y = \tan^{-1}(10x_1 + \sin(5x_2)) + \xi. \quad (2)$$

The influence of x_2 on y when $x_1 = 0$ is almost purely sinusoidal. But when, for example, $x_1 = 1$, x_2 has almost no influence on y . A scatter plot of y versus x_2 (Fig. 1, 1000 random points sampled from two-dimensional uniform distribution with bounds $[-1, +1]$ were used) also does not help to recognize the form of influence.

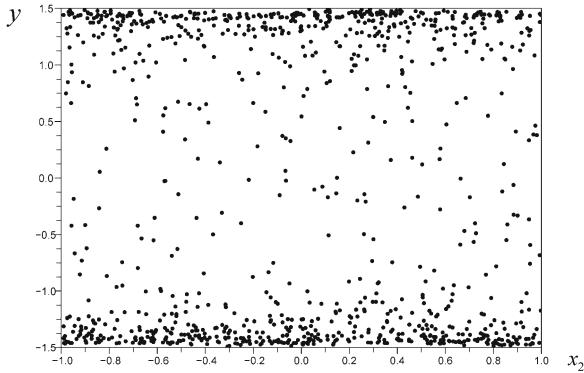


Fig. 1. Scatter plot of function (2)

In a general case, assume that function (1) can be presented as a composition of arbitrary continuous nonlinear functions defined on a compact domain

$$y = f(f_1(x_1), f_2(x_2), \dots, f_n(x_n)) + \xi. \quad (3)$$

It is easy to show that such representation exists for any continuous $g(\cdot)$ because always there exists a special case of (3) where $f(\cdot) \equiv g(\cdot)$, $f_1(x_1) \equiv x_1, \dots, f_n(x_n) \equiv x_n$. Decomposition of function (1) in the form (3) would provide a means of separation of independent influences of input variables described by functions $f_i(x_i)$ from their interdependent influence described by function $f(\cdot)$.

The problem of nonlinear function decomposition has already received attention of researchers, however quite a few solutions with limited capabilities were discovered. A comprehensive review of related literature is given in [8]. Vast majority of the proposed methods concerned only nominally-valued (particularly binary) functions. An attempt to extend these methods to real-valued functions was made in [9], but many limitations (predefined set of basic functions of two variables only, necessity of data discretization, high sensitivity to noise, often dependence on manual interaction in the decomposition process, low parameter estimation accuracy due to discrete nature of the method, etc.) make this method of low practical value.

To overcome these limitations, we employ a different approach based on computational intelligence techniques. Several methods based on neural networks were proposed, which decompose complex functions into a set of simpler functions that are easier to implement [10], [11], [12], but they employ bottom-up approach: subnetworks for smaller functions are built and trained separately, and then aggregated to implement the initial complex function. This approach requires substantial prior knowledge and availability of training data for subnetworks, which often are not available in practice.

In this paper we propose a specialized architecture of artificial neural network called Function Decomposition Network (FDN) that decomposes a real-valued function described by its realization into a set of functions (3) using top-down approach, which requires only training data generated by real $g(\cdot)$ and no prior knowledge. FDN architecture and operation are presented in Section 2, Section 3 contains experimental results, and general discussion is given in Section 4.

2 Function Decomposition Network Architecture

The proposed Function Decomposition Network has a feedforward architecture (Fig. 2). Shapes in circles, which represent neurons, schematically show graphs of their activation functions.

FDN has three hidden layers: the first and the third hidden layers consist of neurons with nonlinear activation functions $\psi^{[1]}$ and $\psi^{[3]}$ respectively (we use hyperbolic tangent for both layers). The second hidden layer and the output layer have neurons with linear activation functions. The first two hidden layers form a set of n subnetworks, which process only their respective inputs. Then their outputs are fed to the third hidden layer, which along with the output layer approximates $f(\cdot)$. Thus the subnetworks act as independent universal approximators, which are trained inside a larger network to approximate independent functions $f_i(x_i)$. The second hidden layer is also called the observation layer. Its function is to extract estimations \hat{f}_i of true signals f_i .

The mapping implemented by FDN can be expressed by

$$\hat{y} = w_0^{[4]} + \sum_{j=1}^{n^{[3]}} \left(w_j^{[4]} \psi^{[3]} \left(w_{j0}^{[3]} + \sum_{i=1}^n w_{ji}^{[3]} \hat{f}_i(x_i) \right) \right), \quad (4)$$

where $n^{[3]}$ is the number of neurons in the third hidden layer, $w_j^{[4]}$ -- weights of the output neuron and $w_0^{[4]}$ -- its bias, $w_{ji}^{[3]}$ -- weights of the j -th neuron of the third hidden layer, $w_{j0}^{[3]}$ -- the corresponding bias.

Functions $f_i(x_i)$ are approximated by corresponding subnetworks implementing the following relations

$$\hat{f}_i(x_i) = w_0^{[2,i]} + \sum_{l=1}^{n^{[1,i]}} \left(w_l^{[2,i]} \psi^{[1]} \left(w_{l0}^{[1,i]} + w_{l1}^{[1,i]} x_i \right) \right), \quad (5)$$

where $n^{[1,i]}$ is the number of neurons in the first hidden layer of the i -th subnetwork, $w_l^{[2,i]}$ – weights of the output neuron of the i -th subnetwork and $w_0^{[2,i]}$ – its bias, $w_{l1}^{[1,i]}$ – a single weight of the l -th neuron of the first hidden layer of the i -th subnetwork, $w_{l0}^{[1,i]}$ – the corresponding bias.

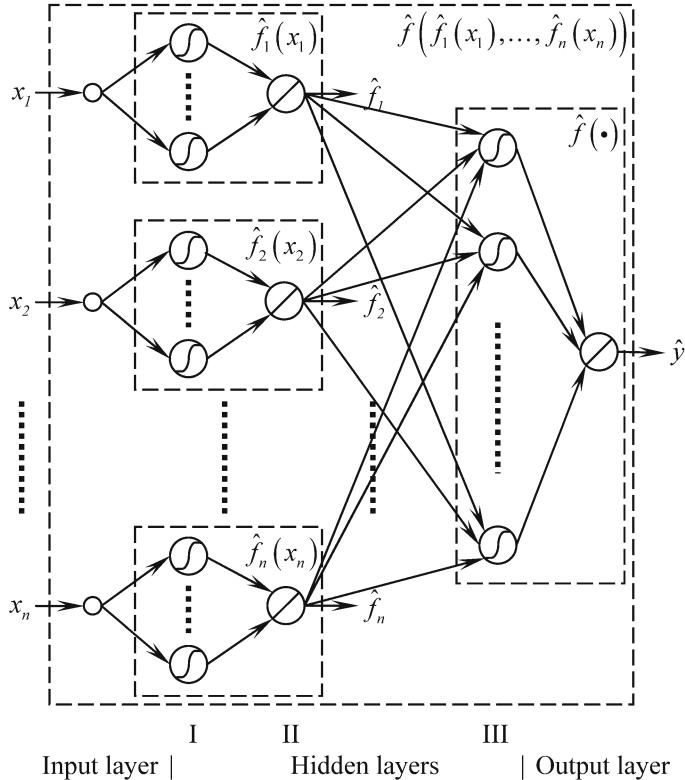


Fig. 2. FDN architecture

At first sight, relation (4) may resemble the formulation of the famous Kolmogorov's superposition theorem [13], whose relevance to feedforward neural networks was first stressed in [14] and then studied by many authors [15–23], however our approach is substantially different. Our approximations $\hat{f}(\cdot)$ and $\hat{f}_i(x_i)$ both depend on $g(\cdot)$, moreover $\hat{f}_i(x_i)$ are not limited to be monotonic but are intended to catch true dependencies between x_i and y , and may have any shape.

The network is utilized in the following manner. First, available data pairs $\{(x_1(k), x_2(k), \dots, x_n(k)), y(k)\}, k = 1, \dots, N$ (here N is the data set size) are used to train the network. The main requirement for the training set is to be representative, i.e. to cover the whole range of possible values of x_i .

FDN is trained with respect to the standard quadratic criterion applied to the error signal $e(k) = y(k) - \hat{y}(k)$:

$$E = \sum_{k=1}^N e^2(k) = \sum_{k=1}^N (y(k) - \hat{y}(k))^2. \quad (6)$$

As far as the network has feedforward architecture, standard backpropagation procedure can be applied to compute local gradients. Then Levenberg-Marquardt algorithm with cross-validation in batch mode is applied to tune network weights. This topic is well covered in the literature [24], so we will not focus on it.

Observation outputs \hat{f}_i are not utilized during training phase, since there are no reference signals for them. For the same reason, the estimated functions $\hat{f}_i(x_i)$ are subject to arbitrary linear transformations (sign reversal, scaling, and shifting). It is therefore convenient for further analysis to change the sign of the observed signals \hat{f}_i as necessary and scale to the range $[0, 1]$ by applying the following transformation

$$\tilde{f}_i = \frac{\hat{f}_i - \hat{f}_{i,\min}}{\hat{f}_{i,\max} - \hat{f}_{i,\min}}, \quad (7)$$

where $\hat{f}_{i,\min}$ and $\hat{f}_{i,\max}$ are the minimal and maximal values of \hat{f}_i respectively.

Then the trained subnetworks are extracted from FDN and used independently as approximators of the corresponding $f_i(x_i)$. There exist a number of ways to utilize these approximators. The simplest one is to use them for visual analysis of plots $\hat{f}_i(x_i)$, which are obtained by feeding linearly growing signal $x_i(k), x_{i,\min} \leq x_i(k) \leq x_{i,\max}$ ($x_{i,\min}$ and $x_{i,\max}$ are the minimal and maximal values of x_i respectively) to the input of the corresponding subnetwork and observing the output $\hat{f}_i(x_i(k))$. It is also possible to study the first and the second differences of $\hat{f}_i(x_i(k))$ to find singular points (extrema and inflection points). Other types of analysis are also applicable.

3 Experimental Results

Abilities of our approach are illustrated by the following simulations based on artificially constructed functions, which allow decomposition error for all \hat{f}_i to be calculated. For convenience of analysis, all test functions are defined on a hypercube $x_i \in [-1, +1]$. Data sets are generated by randomly sampling 10000 points from n -dimensional uniform distribution bounded by the abovementioned hypercube and computing the test functions at those points. 90% of data is used for network training, and 10% for out-of-sample testing. The noise term ξ in all experiments is a normally distributed random variable with zero mean and standard deviation 0.01. The number of neurons in the hidden layers of FDN is $n^{[1,i]} = 10$ and $n^{[3]} = 25$. After training, function approximations $\hat{f}_i(x_i)$ are extracted by applying linearly growing signal to the corresponding network inputs and observing the output signals \hat{f}_i . True test functions $f_i(x_i)$ and their

approximations $\hat{f}_i(x_i)$ are both scaled to the interval $[0, 1]$ so that it is possible to compare them and calculate approximation errors.

We start with a simple function

$$y = \cos(x_1 + x_2 + x_3) + \xi. \quad (8)$$

The normalized root mean square error (NRMSE) on the training set is 0.022954, on the test set 0.024684. Graphs of true functions $f_i(x_i)$ and their FDN approximations $\hat{f}_i(x_i)$ are plotted in Fig. 3 (in all graphs solid line represents true function and dashed line – approximation). Approximation errors are: $\text{NRMSE}_1 = 0.006565$, $\text{NRMSE}_2 = 0.007777$, $\text{NRMSE}_3 = 0.008176$. We can see that the network easily solved this decomposition problem.

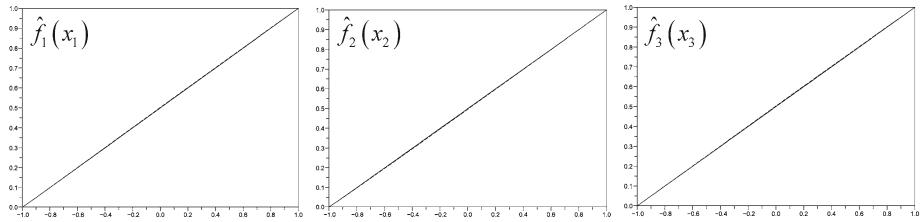


Fig. 3. Graphs of true functions and their FDN approximations for function (8)

Then we return to our introductory example (2). NRMSE on the training set is 0.007474, on the test set 0.010682. Graphs of true functions and their FDN approximations are plotted in Fig. 4. Approximation errors are: $\text{NRMSE}_1 = 0.110080$, $\text{NRMSE}_2 = 0.009599$. As we can see, the task that seemed to be hard after looking at Fig. 1 now is solved by FDN with a reasonable accuracy.

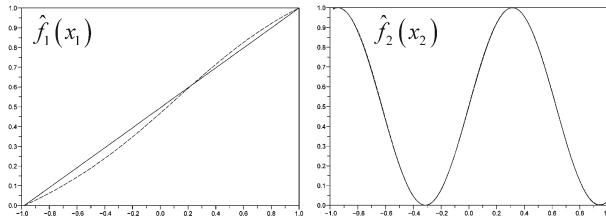


Fig. 4. Graphs of true functions and their FDN approximations for function (2)

Now consider a more complex function, which involves multiplication,

$$y = \sin(5x_1)x_2^2 + 5x_3 + \xi. \quad (9)$$

NRMSE on the training set is 0.003518, on the test set 0.003558. Graphs of true functions and their FDN approximations are plotted in Fig. 5. Approximation errors are: $\text{NRMSE}_1 = 0.015760$, $\text{NRMSE}_2 = 0.362474$, $\text{NRMSE}_3 = 0.029165$. Here the network had some problems with the parabolic function, however the shape is still very close to the original and can be easily recognized by visual examination.

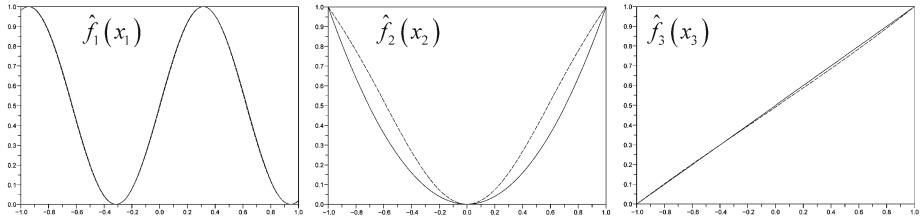


Fig. 5. Graphs of true functions and their FDN approximations for function (9)

In the next example the network has to distinguish between dependent (x_1x_3) and independent ($\tan(1.5x_1)$, x_3) influences of x_1 and x_3 :

$$y = \sin(\tan(1.5x_1) + x_2^2 + x_3 + x_1x_3) + \xi. \quad (10)$$

NRMSE on the training set is 0.112799, on the test set 0.202078. Graphs of true functions and their FDN approximations are plotted in Fig. 6. Approximation errors are: $\text{NRMSE}_1 = 0.396391$, $\text{NRMSE}_2 = 0.035662$, $\text{NRMSE}_3 = 0.056657$. Here tangent function is approximated not very accurately, but the overall shape is correct and, what's more important, the x_1x_3 term did not deteriorate the independent terms. As we can see, the solution of test problems proved good decomposition capabilities of the proposed network. This result is also supported by a real-world problem solution.

In [25] we reported on the method for extracting the temperature-load relationship from historic electric load and weather data sets using a modification of the FDN architecture. The method was successfully applied for analysis of two

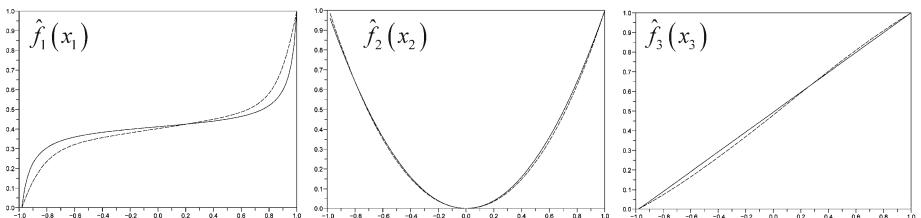


Fig. 6. Graphs of true functions and their FDN approximations for function (10)

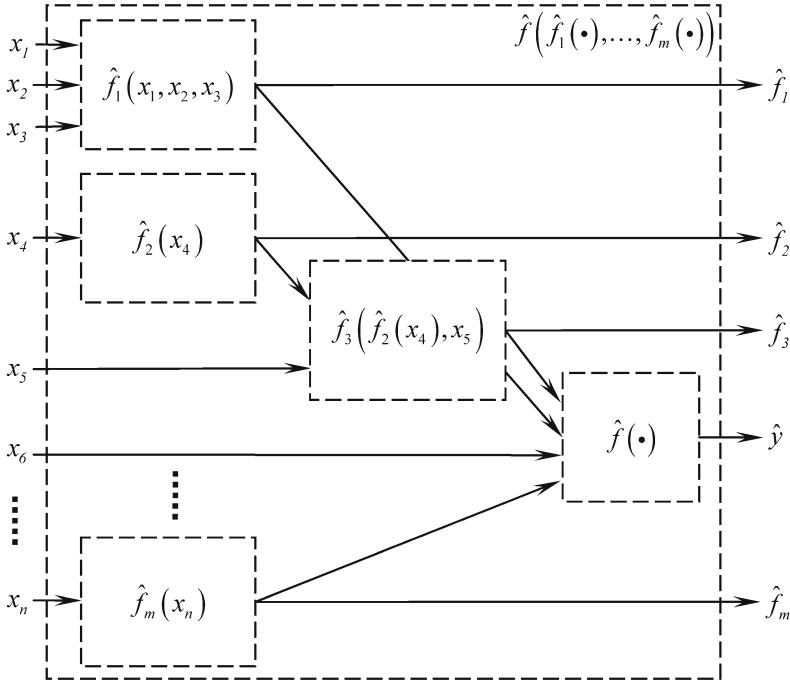


Fig. 7. FDN architecture corresponding to decomposition (II)

power systems located in different parts of the world (USA and Ukraine) with substantially different climatic conditions. The revealed relationships conformed to the theoretically justified temperature-load curve. Analysis of the first and the second differences provided exact values of extreme and inflection points of the revealed temperature-load dependencies. We also showed that several studies reported in literature used incorrect approximations of temperature-load relationships, which could lead to higher errors in electric load forecasting problems. Further analysis of the revealed relationships provides electric companies with valuable information for more accurate electric load and power loss forecasting, prospective power systems planning, and facilitates more efficient utilization of power grids and systems.

4 Discussion

FDN architecture can be seen as a modular one with n modules formed by the first two hidden layers and another module formed by the last two layers. But in contrast to usual applications of modular networks, where each module is constructed and trained independently for its specific task and afterwards they are combined together, in our approach, the network is treated as monolithic, and only after training parts of it may be used independently as modules. These

trained subnetworks can be used not only for input influence analysis, but also for the signal preprocessing, forecasting, extrapolation and other tasks.

In this paper we described the most general case without any prior knowledge about the structure of function $g(\cdot)$. However in many practical applications such information may be available to some extent, or may be inferred from the training data [26]. It can be easily incorporated in the FDN architecture by choosing appropriate structure for subnetworks and connections between them. For example, if prior knowledge suggests the following form of decomposition

$$y = f(f_1(x_1, x_2, x_4), f_3(f_2(x_4), x_5), x_6, \dots, f_m(x_n)) + \xi, \quad (11)$$

then it is appropriate to utilize FDN with architecture shown in Fig. 7. Notice, that multilevel (hierarchical) decomposition is possible, and in any case approach remains top-down: train the whole FDN using available realizations of $g(\cdot)$, then extract subnetworks for further independent utilization.

Despite all the positive results, we have to note that decomposition (3) may not be unique for any given function (1), however when learning error of FDN in our experiments with presented and other test functions was low (NRMSE is in the order of 10^{-2} or below), the subnetworks were always able to approximate true independent functions. Still this question needs further investigation and will be a topic of our future research.

References

1. Nelles, O.: Nonlinear System Identification. Springer, Berlin (2001)
2. Haykin, S.: Neural Networks. A Comprehensive Foundation. Prentice Hall, Upper Saddle River (1999)
3. Pham, D.T., Liu, X.: Neural Networks for Identification, Prediction and Control. Springer, London (1995)
4. Jang, J.-S.R., Sun, C.-T., Mizutani, E.: Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Prentice Hall, Upper Saddle River (1997)
5. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. Systems, Man, and Cybernetics* 15, 116–132 (1985)
6. Yamakawa, T., Uchino, E., Miki, T., Kusanagi, H.: A neo-fuzzy neuron and its applications to system identification and prediction of the system behavior. In: Proc. 2nd Int. Conf. Fuzzy Logic and Neural Networks, Iizuka, Japan, pp. 477–483 (1992)
7. Tikk, D., Koczy, L.T., Gedeon, T.D.: A survey on universal approximation and its limits in soft computing techniques. *International Journal of Approximate Reasoning* 33, 185–202 (2003)
8. Zupan, B., Bratko, I., Bohanec, M., Demsar, J.: Function Decomposition In Machine Learning. In: Palioras, G., Karkaletsis, V., Spyropoulos, C.D. (eds.) ACAI 1999. LNCS (LNAI), vol. 2049, pp. 71–101. Springer, Heidelberg (2001)
9. Demsar, J., Zupan, B., Bohanec, M., Bratko, I.: Constructing intermediate concepts by decomposition of real functions. In: van Someren, M., Widmer, G. (eds.) ECML 1997. LNCS, vol. 1224, pp. 93–107. Springer, Heidelberg (1997)

10. Vaccari, D.A., Wojciecliowski, E.: Neural Networks as Function Approximators: Teaching a Neural Network to Multiply. Proc. IEEE World Congress on Computational Intelligence 4, 2217–2221 (1994)
11. Lu, X.-J., Li, H.-X.: Sub-domain intelligent modeling based on neural networks. Proc. IEEE World Congress on Computational Intelligence, 445–449 (2008)
12. Jansen, W.J., Diepenhorst, M., Nijhuis, J.A.G., Spaanenburg, L.: Assembling Engineering Knowledge in a Modular Multi-layer Perception Neural Network. In: Proc. Int. Conf. on Neural Networks, vol. 1, pp. 232–237 (1997)
13. Kolmogorov, A.N.: On the representation of continuous function of many variables by superpositions of continuous functions of one variable and addition. Doklady Akademii Nauk USSR 114, 953–956 (1957) (in Russian)
14. Hecht-Nielsen, R.: Kolmogorov's mapping neural network existence theorem. In: Proc. of the International Conference on Neural Networks, vol. III, pp. 11–14. IEEE Press, New York (1987)
15. Girosi, F., Poggio, T.: Representation properties of networks: Kolmogorov's theorem is irrelevant. Neural Computation 1, 465–469 (1989)
16. Kurkova, V.: Kolmogorov's Theorem Is Relevant. Neural Computation 3, 617–622 (1991)
17. Kurkova, V.: Kolmogorov's theorem and multilayer neural networks. Neural Networks 5, 501–506 (1992)
18. Katsura, H., Sprecher, D.A.: Computational Aspects of Kolmogorov's Superposition Theorem. Neural Networks 7, 455–461 (1994)
19. Gorban, A.N., Wunsch, D.C.: The General Approximation Theorem. Proc. IEEE World Congress on Computational Intelligence 2, 1271–1274 (1998)
20. Neruda, R., Stedry, A., Drkosova, J.: Kolmogorov learning for feedforward networks. In: Proc. International Joint Conference on Neural Networks, vol. 1, pp. 77–81 (2001)
21. Sprecher, D.A., Draghici, S.: Space-filling curves and Kolmogorov superposition-based neural networks. Neural Networks 15, 57–67 (2002)
22. Igelnik, B., Parikh, N.: Kolmogorov's Spline Network. IEEE Trans. Neural Networks 14, 725–733 (2003)
23. Kolodyazhniy, V., Bodyanskiy, Y.: Fuzzy Kolmogorov's network. In: Proc. 8th Int. Conf. Knowledge-Based Intelligent Information and Engineering Systems, Wellington, New Zealand, pp. 764–771 (2004)
24. Shepherd, A.J.: Second-Order Methods for Neural Networks (Fast and Reliable Training Methods for Multi-Layer Perceptrons). Springer, London (1997)
25. Bodyanskiy, Y., Popov, S., Rybalchenko, T.: Feedforward Neural Network with a Specialized Architecture for Estimation of the Temperature Influence on the Electric Load. In: Proc. 4th Int. IEEE Conf. Intelligent Systems, Varna, Bulgaria, vol. I, pp. 7-14-17-18 (2008)
26. Chambliss, B., Lendaris, G.G., Zwick, M.: An Information Theoretic Methodology for Prestructuring Neural Networks. In: Proc. International Joint Conference on Neural Networks, vol. 1, pp. 365–369 (2001)

Improved Storage Capacity in Correlation Matrix Memories Storing Fixed Weight Codes

Stephen Hobson and Jim Austin

Advanced Computer Architectures Group
Department of Computer Science
University of York
Heslington
York, YO10 5DD, UK
`{stephen,austin}@cs.york.ac.uk`

Abstract. In this paper we introduce an improved binary correlation matrix memory (CMM) with better storage capacity when storing sparse fixed weight codes generated with the algorithm of Baum et al. [3]. We outline associative memory, and describe the binary correlation matrix memory—a specific example of a distributed associative memory. The importance of the representation used in a CMM for input and output codes is discussed, with specific regard to sparse fixed weight codes. We present an algorithm for generating of fixed weight codes, originally given by Baum et al. [3]. The properties of this algorithm are briefly discussed, including possible thresholding functions which could be used when storing these codes in a CMM; L-max and L-wta. Finally, results generated from a series of simulations are used to demonstrate that the use of L-wta as a thresholding function provides an increase in storage capacity over L-max.

Keywords: Associative memory, correlation matrix memory, storage capacity, fixed weight codes, pattern recognition.

1 Introduction

The fixed weight code generation algorithm of Baum et al. [3] has the benefit of generating unique codes which are well separated in the pattern space, which makes them suited for storage in a CMM. We will subsequently term the codes generated from this algorithm *Baum codes*, for ease of reference. Recall of such codes from a CMM requires the use of a thresholding function, and it is with the nature of this function that this paper is concerned. L-max thresholding [1] has been shown to be an effective thresholding function for fixed weight codes, and has been applied to CMMs storing Baum codes [2]. However, L-max thresholding fails to make use of all the constraints on Baum codes. A thresholding mechanism which takes advantage of these constraints is able to provide an improved storage capacity for a CMM which stores Baum codes.

2 Associative Memory

Traditional computer memories store data in a compartmentalised fashion, with each stored item having a unique address. While this leads to perfect recall in all cases where the correct address is known, any amount of error in the address will result in a recall which bears no relation to the stored item. A contrasting model of memory is *distributed associative memory*. In such a model, data items are stored as pairs, with the presentation of the first member of the pair to the memory resulting in the recall of the second. Rather than these associations being stored in a single location in the memory, they are distributed across the memory. This provides robustness in the presence of noise on the input, and enables generalisation.

Such a memory serves a different purpose to a traditional memory. While a standard computer memory is well suited to tasks such as storing a list of tasks or events, it is less capable of “answering questions” [6]. Such a task would require the question to be looked up in a list, which might contain the location of the answer. In a distributed associative memory, the answer is retrieved simply by presenting the question to the input. The recall operation does not require a look-up algorithm, and so is a much more efficient operation.

3 Binary Correlation Matrix Memories

A Binary Correlation Matrix Memory (CMM) [8] is one example of a distributed associative memory. It stores the correlations between input and output codes. The memory is a one layer fully connected neural network. This means that the weights can be viewed as an $m \times n$ matrix W , where m is the size of the input vectors and n is the size of the output vectors. An example of such a memory is shown in Fig. 1. Although it is possible to use a CMM with non-binary weights [5], only the binary case will be considered in this paper.

Learning is achieved using an outer product of the input and output. These matrices are combined using an OR function over all input output pairs to create the weight matrix W .

$$W = \bigvee_{i=1}^N x_i y_i^T \quad (1)$$

Recall is achieved as shown in Equation 2.

$$y = f[Wx] \quad (2)$$

Here f is a thresholding function, which takes the activity output Wx and converts it to a binary vector. For example, in their original paper Willshaw et al. [8] suggested that the thresholding function could set all output nodes with activity greater than the number of 1s in the input pattern x to 1. The choice of this threshold function has a profound effect on the storage capability of the network, as we shall see later.

When recalling a pattern from the memory, the resulting vector (before thresholding) can be viewed as a signal (the original stored pattern) and some noise (extra activity from overlaps with other learned codes).

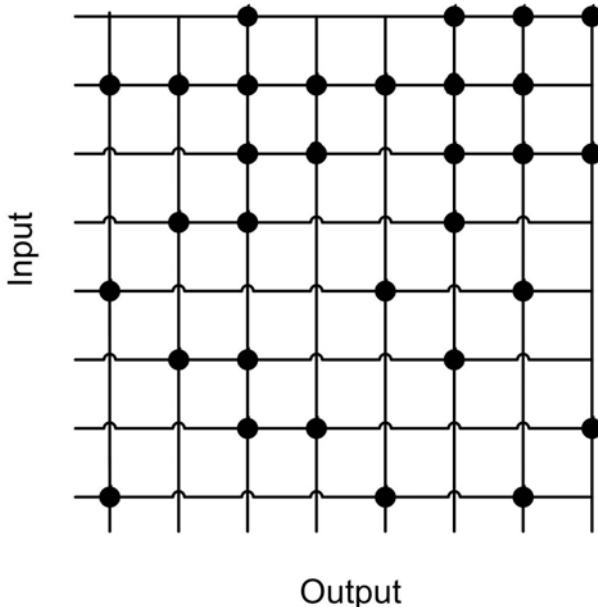


Fig. 1. An example of a CMM with 8 input neurons, 8 output neurons and binary weights

4 Sparse Fixed Weight Coding

In a neural memory such as a CMM there is an intrinsic link between the data representation used and the storage capability of the memory. Using different encodings on the input and output of the memory will have different effects on the storage capacity. In addition, the choice of threshold function will also relate directly to the representation used.

Perhaps the simplest representation of all would be the use of unary output codes (a single bit set to 1 in n bits). This provides a storage capacity of exactly n code pairs. Each input code will be stored in exactly one column of the matrix, and given a correct input code there will be no error on recall. However, the fault tolerance capability of the network is lost, since the storage is no longer distributed. It is necessary to use input and output codes with more than one bit set to 1 to distribute storage over the network.

Furthermore, in order to maximise the storage capability of the network, these codes should be sparsely coded. More specifically, the number of 1s in an n bit code should be in the order of $\log n$ [7]. The number of 1s in an n bit code is termed the *weight* of the code. An important property of the use of sparse codes in a CMM is that the memory is capable of storing $k > n$ codes (where k is the number of pairs stored, and n is the number of input neurons), providing a small amount of recall error is tolerated [7].

If codes with fixed weight are used, an alternative threshold function is available; L-max thresholding [1]. This sets the l neurons with the highest output

1	0	0	0	0	0	1	0	0	1
0	1	0	0	0	0	1	0	0	1
0	0	1	0	0	0	1	1	0	
0	0	0	1	0	1	0	0	0	1
0	0	0	0	1	0	1	0	1	0
1	0	0	0	0	0	0	1	0	1
0	1	0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1	0	
0	0	0	1	1	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Fig. 2. An example of the generation of Baum codes. Here, $l = 3$ and $p_1 = 5, p_2 = 3, p_3 = 2$, giving a code of length 10.

activity to 1 and the rest to 0, where l is the weight of the output code. Casasent and Telfer [4] experimented with various output encodings, including Binary codes, Hamming codes and fixed weight codes, albeit with analog input codes. They found that in the presence of noise, fixed weight codes with L-max thresholding gave the greatest storage capacity for a given code length.

It is important to have the ability to generate fixed weight codes in such a fashion that the codes generated are guaranteed to be well separated in pattern space. Baum et al. proposed an algorithm which generates fixed weight codes which have a small amount of overlap [3]. The code is divided into l sections which are relatively prime¹ (coprime) in length, with each section i having length p_i . For example, a code of length 32 where $l = 3$ could be divided into sections of length 16, 9 and 7. The size of l defines the weight of the code. To generate code number c , we set the bit in position j as follows (where x is the code to output):

$$x_j^c = \begin{cases} 1 & \text{if } j - \sum_{k=1}^{i-1} p_k \equiv c \pmod{p_i} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Essentially what is happening is that a single bit will be set to 1 in each section of the code. As subsequent codes are generated, the next bit in each section will be set to 1 instead, wrapping around to the beginning of the section when the end is reached. For example, Fig. 2 shows a code with $n = 10$ and $l = 3$, taking $p_1 = 5, p_2 = 3, p_3 = 2$.

Using this mechanism $p_1 \times p_2 \times \dots \times p_s$ unique codes can be generated, which is substantially fewer than it is possible to represent with a general fixed weight coding scheme ($\frac{n!}{(n-l)!l!}$). However, the overlap between the codes is guaranteed

¹ Two integers are relatively prime if they have no common factor other than 1. It should be noted that the problem of generating a set of relatively prime numbers which sum to a total is not trivial. However, a discussion of methods is beyond the scope of this paper.

to be small, which improves recall accuracy if they are used in a CMM. Since the method is deterministic, we can be certain about the amount of overlap between generated codes. With no loss of generality we can consider a Baum code with section lengths $p_1 < p_2 < \dots < p_t$. The first p_1 codes generated will have no overlap at all. The first $p_1 p_2$ overlap by at most 1 bit (a Hamming distance of at least $2l - 2^t$). In their analysis Baum et al. [3] state that if $\prod_{i=1}^t p_i$ codes are used, the minimum Hamming distance between any two codes will be $d = 2(l - t + 1)$. For this reason, it is beneficial for $p_i \approx n/l$, since this maximises the product between the section lengths p_i , and hence the number of codes which can be generated with minimal overlap.

5 Improving the Storage Capacity

In the past the algorithm of Baum et al. [3] has been used to generate fixed weight codes, with L-max used as the thresholding function [2]. This represents an oversight, since L-max thresholding may produce output codes which are not possible under the Baum algorithm. By constraining the threshold so that only the codes generated by the algorithm are output, an increased storage capacity can be achieved.

It has already been mentioned that the algorithm divides the code into a series of sections. Baum et al. point out in the appendix to their paper that a useful property of the algorithm is that there is exactly one 1 in each section of the code. This means that a winner-takes-all (WTA) threshold can be applied to each section of the code, rather than taking the l highest values from the whole code, as we would with L-max [3]. This thresholding technique incorporates more information about the output encoding into the thresholding function, and therefore provides a more robust thresholding. We shall call this thresholding technique L-wta.

6 Results

To demonstrate the improved storage capacity of a CMM when using L-wta compared to L-max a series of simulations were conducted. The storage of a CMM is affected by the size of the input and output codes, and also by the weight of the coding system used. For this reason L-wta technique was compared to L-max for a variety of coding systems. In each experiment an empty CMM was created for the appropriate code sizes. The following steps were then undertaken.

1. Generate an input code according to the algorithm of Baum et al. [3] This code will be unique.
2. Generate a random output code. This code is a random code from the entire space of possible Baum codes for the given set of coprimes, and so may not be unique.

² The Hamming distance between two codes is the number of bits which differ between them.

Table 1. Experimental results when varying the size of the input code. All tables show the number of codes learnt before errors at given levels. Codes are given in the format *length/weight*. Note that in some cases code lengths are approximate. This is due to the complexity of generating large sets of coprime numbers which sum to a given target. Bold numbers show the percentage increase in storage capacity when using L-wta rather than L-max.

Input code	Output code	0.1% error		1% error		5% error		10% error		
		L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	
64/4	256/4	70	78	11.4	115	129	12.2	153	171	11.8
128/4	256/4	139	141	1.4	197	234	18.8	289	334	15.6
256/4	256/4	259	286	10.4	424	473	11.6	600	696	16.0
512/4	256/4	496	589	18.8	814	927	13.9	1182	1369	15.8
								1416	1630	15.1

Table 2. Experimental results when varying the weight of the input code

Input code	Output code	0.1% error		1% error		5% error		10% error		
		L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	
512/2	256/4	260	260	0.0	282	304	7.8	482	555	15.1
512/4	256/4	496	589	18.8	814	927	13.9	1182	1369	15.8
512/8	256/4	1023	1036	1.3	1453	1603	10.3	1811	1989	9.8
512/16	256/4	1186	1267	6.8	1408	1512	7.4	1673	1824	9.0
								1829	1989	8.7

Table 3. Experimental results when varying the size of the output code

Input code	Output code	0.1% error		1% error		5% error		10% error		
		L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	
256/4	64/4	91	120	31.9	178	203	14.0	245	286	16.7
256/4	128/4	148	179	20.9	265	289	9.1	362	429	18.5
256/4	256/4	259	286	10.4	424	473	11.6	600	696	16.0
256/4	512/4	373	415	11.3	608	712	17.1	950	1099	15.7
								1137	1327	16.7

Table 4. Experimental results when varying the weight of the output code

Input code	Output code	0.1% error		1% error		5% error		10% error		
		L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	L-max	L-wta %	
256/4	512/2	564	709	25.7	1259	1435	14.0	1932	2138	10.7
256/4	512/4	373	415	11.3	608	712	17.1	950	1099	15.7
256/4	512/8	257	267	3.9	353	400	13.3	495	569	14.9
256/4	512/16	71	89	25.4	138	157	13.8	197	222	12.7
								219	266	21.5

3. Train the CMM using the generated input/output pair.
4. Present every input which the CMM has learnt and compare the correct output to the actual output using L-max and L-wta.
5. If average error (defined below) exceeds 10% for all thresholding techniques then exit, otherwise return to 1.

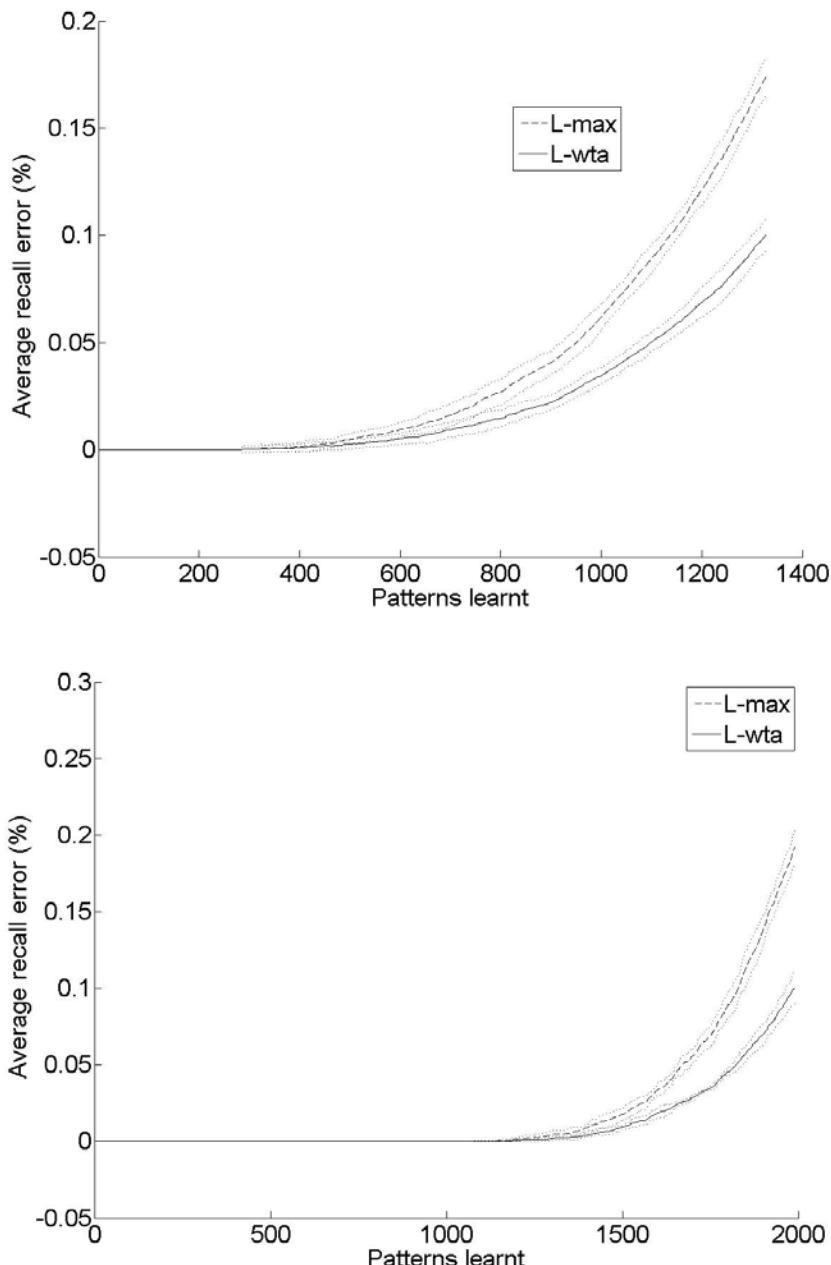


Fig. 3. Two comparisons of the storage capabilities of a CMM when using L-max and L-wta. Dotted lines show the standard deviation of average recall error between runs of the experiment. (top) Input codes had size 256 and weight 4. Output codes had size 512 and weight 4. (bottom) Input codes had size 512 and weight 16. Output codes had size 256 and weight 4.

For each experiment these steps were run with twenty CMMs. A different integer was used to seed the random generator for each CMM, resulting in differing output patterns being trained. After each iteration the average error was calculated for all twenty CMMs. The recall error was defined as the percentage of recalled patterns which contained an error in any bit. In order to measure the performance of the thresholding techniques at a variety of error tolerances we examine the number of codes learnt in each memory before recall error exceeded 0.1%, 1%, 5% and 10%.

Table 1 shows the results when the size of the input was varied, whilst size and weight of the output code remained constant. Similarly, Table 2 shows the results when the weight of the input code was varied. In both cases it can be seen that the use of L-wta results in an increase of approximately 15% in storage capacity. L-wta appears to provide the largest increase in storage over L-max when the input code is sparse; that is, when the code size is increased or the weight is decreased. This advantage appears less pronounced as the amount of output error increases.

The case is similar when examining Tables 3 and 4. The effect of output sparsity on the effectiveness of the technique is less clear. However, the storage capacity when using L-wta is consistently a considerable improvement over that achieved using L-max.

Fig. 3 shows two examples of the performance of the two thresholding techniques as codes are trained into the memories. It can clearly be seen that as the memory becomes increasingly saturated, the use of L-wta provides an increasing benefit over L-max thresholding.

7 Summary

In summary, it has been demonstrated in this paper that when using codes generated by the algorithm of Baum et al. [3] L-wta provides an increase in storage capacity over thresholding using L-max, provided some error is tolerated. This increase in storage capacity is generally in the order of 15%, but has been observed to be as high as 30%.

While this paper has demonstrated the benefit of this thresholding technique across a variety of conditions, a mathematical treatment of the technique is now required.

References

1. Austin, J., Stonham, T.: Distributed associative memory for use in scene analysis. *Image and Vision Computing* 5, 251–260 (1987)
2. Austin, J., Kennedy, J.V., Lees, K.: A neural architecture for fast rule matching. In: *Artificial Neural Networks and Expert Systems Conference* (1995)
3. Baum, E.B., Moody, J., Wilczek, F.: Internal representations for associative memory. *Biological Cybernetics* 59(4), 217–228 (1988)
4. Casasent, D., Telfer, B.: High capacity pattern recognition associative processors. *Neural Networks* 5, 687–698 (1992)

5. Nadal, J., Toulouse, G.: Information storage in sparsely coded memory nets. *Network* 1, 61–74 (1990)
6. Palm, G.: Neural Assemblies. Springer, Heidelberg (1982)
7. Palm, G., Schwenker, F., Sommer, F.T., Strey, A.: Neural associative memories. *Associative processing and processors*, 307–326 (1997)
8. Willshaw, D.J., Buneman, O.P., Longuet-Higgins, H.C.: Non-holographic associative memory. *Nature* 222(7), 960–962 (1969)

Multiagent Reinforcement Learning with Spiking and Non-Spiking Agents in the Iterated Prisoner’s Dilemma

Vassilis Vassiliades, Aristodemos Cleanthous, and Chris Christodoulou

Department of Computer Science, University of Cyprus,
75 Kallipoleos Ave., P.O. Box 20537, 1678 Nicosia, Cyprus
`{v.vassiliades,aris,cchrist}@cs.ucy.ac.cy`

Abstract. This paper investigates Multiagent Reinforcement Learning (MARL) in a general-sum game where the payoffs’ structure is such that the agents are required to exploit each other in a way that benefits all agents. The contradictory nature of these games makes their study in multiagent systems quite challenging. In particular, we investigate MARL with spiking and non-spiking agents in the Iterated Prisoner’s Dilemma by exploring the conditions required to enhance its cooperative outcome. According to the results, this is enhanced by: (i) a mixture of positive and negative payoff values and a high discount factor in the case of non-spiking agents and (ii) having longer eligibility trace time constant in the case of spiking agents. Moreover, it is shown that spiking and non-spiking agents have similar behaviour and therefore they can equally well be used in any multiagent interaction setting. For training the spiking agents, a novel and necessary modification enhances competition to an existing learning rule based on stochastic synaptic transmission.

1 Introduction

Multiagent Reinforcement Learning (MARL) is a problem that has been studied extensively during the last few years. The problem lies in the dynamic environment created by the presence of another learner. In MARL there could be different kinds of situations: fully competitive (which could be modelled with zero-sum games), fully cooperative (which could be modelled with team games) or a mixture of both (which could be modelled with general-sum games). Each situation has different problems, so an active community has been designing algorithms to address all of them. Some examples include minimax-Q [1], Nash-Q [2], Joint Action Learners [3], FoF-Q (Friend-or-Foe Q) [4], WoLF-IGA (Win or Lose Fast - Infinitesimal Gradient Ascent) [5], CE-Q (Correlated Equilibria Q) [6], FMQ (Frequency Maximum Q) [7], GIGA-WoLF (Generalised IGA - WoLF) [8] and AWESOME (Adapt When Everybody is Stationary Otherwise Move to Equilibrium) [9].¹ A lot of work is focused in deriving theoretical guarantees, based on different sorts of criteria such as rationality and convergence [11] or

¹ For a comprehensive coverage of MARL algorithms see [10] and references therein.

targeted-optimality, safety and auto-compatibility [12]. Since the problem is not very well-defined, Shoham et al. [13] attempted to classify the existing work by identifying five distinct research agendas. They argued that when researchers design algorithms they need to place their work under one of these categories. Subsequently some work did focus on specific agendas (e.g., [14]), but more agendas were proposed [15]. In addition, the original agendas [13] have been criticised that they may not be distinct, since they may complement each other ([16], [17]). Stone [18] extended the criticism by arguing that the game theoretic approach is not appropriate in complex multiagent problems. Despite these criticisms, our study lies in the original prescriptive non-cooperative agenda [13], which asks how the agents should act to obtain high rewards for a given environment.

Reinforcement learning (RL) has successfully been applied to spiking neural networks (NNs) in recent years. These techniques try to incorporate reward distribution according to the biological processes of neurons. Although their degree of experimental justification varies and they need to be further assessed, all these methods are biologically plausible and provide the basis for applying RL on biologically realistic neural models as well as the inspiration for further and better integration of RL into the spiking models. Imaginative approaches to the subject include: (i) the reinforcement of irregular spiking [19], where the learning rule performs stochastic gradient ascent on the expected reward by correlating the fluctuations in irregular spiking with a reward signal and (ii) the employment of a spike-timing-dependent synaptic plasticity rule [20], in order to achieve RL by modulating this plasticity through reinforcement signals ([21],[22]). In our study, we use a variation of Seung's RL on spiking NNs [23], which reinforces the stochasticity present in the process of synaptic transmission. To the best of our knowledge, this is the first time that a spiking neural model with biologically plausible learning simulates a game theoretical situation.

The current study investigates cooperation between self-seeking reward agents in a non-cooperative setting. This situation can be modelled with the Iterated Prisoner's Dilemma (IPD) which is a general-sum game. Although the cooperative outcome is a valid equilibrium of the IPD, our study does not aim to assess the strength of the learning algorithms to attain equilibria of the game or best responses to any given strategy. Instead, we focus on mutual cooperation and see whether it can be achieved by spiking and simple non-spiking agents trained with RL and attempt to compare them. It is very interesting and beneficial to understand how and when cooperation is achieved in the IPD's competitive and contradictive environment, as it could then become possible to prescribe optimality in real life interactions through cooperation, analogous to the IPD. In its standard one-shot version, the Prisoner's Dilemma [24] is a game summarized by the payoff matrix of Table I. There are 2 players, Row and Column. Each player has the choice of either to "Cooperate" (C) or "Defect" (D). For each pair of choices, the payoffs are displayed in the respective cell of the payoff matrix of Table I. In game theoretical terms, where rational players are assumed, DD is the only Nash equilibrium outcome [25], whereas only the cooperative (CC) outcome satisfies Pareto optimality [26]. The "dilemma" faced by the players in any

Table 1. Payoff matrix of the Prisoner’s Dilemma game with the values used in our experiments. Payoff for the Row player is shown first. R is the “reward” for mutual cooperation. P is the “punishment” for mutual defection. T is the “temptation” for unilateral defection and S is the “sucker’s” payoff for unilateral cooperation. The only condition imposed to the payoffs is that they should be ordered such that $T > R > P > S$.

	Cooperate (C)	Defect (D)
Cooperate (C)	R(=4),R(=4)	S(=-3),T(=5)
Defect (D)	T(=5),S(=-3)	P(=-2),P(=-2)

valid payoff structure is that, whatever the other does, each one of them is better off by defecting than cooperating. The outcome obtained when both defect however is worse for each one of them than the outcome they would have obtained if both had cooperated. In the IPD, an extra rule ($2R > T + S$) guarantees that the players are not collectively better off by having each player alternate between C and D, thus keeping the CC outcome Pareto optimal. Moreover, contrary to the one shot game, CC can be a Nash equilibrium in the infinite version of the IPD.

As pointed out in [27] “perfectly predicting the environment is not enough to guarantee good performance”, because the level of performance depends partly on properties of the environment. In our case, we believe that the property of the environment which plays a significant role in the CC outcome is the reward function, since it specifies the type and strength of the reinforcement the agents receive. By experimenting with the payoff matrix we observed that for the non-spiking agents it is beneficial to mix positive and negative values. The payoff values used in both spiking and non-spiking simulations are shown in Table 1.

The remainder of the paper is organised as follows. Section 2 describes our methodology for both spiking and non-spiking simulations. The results are presented and analysed in Section 3, while the last section gives the conclusions.

2 Methodology

2.1 Agents as Lookup Tables and Non-Spiking Neural Networks

We mainly focus on Q-learning [28], a simple reinforcement learning algorithm. There are two ways of storing the Q values (i.e., the estimates of how good the state-action pairs are), either in a lookup table (LT) or inside a function approximator, such as a NN. When the search space is small, LTs are preferred, because they are much faster. However, for large search spaces, function approximators are the only choice, because it is impossible to store every possible state-action pairs, due to memory constraints. According to Sandholm and Crites [29], LTs yield better results and are faster than simple recurrent NNs in the IPD. The multiagent system can be either *homogeneous* or *heterogeneous*. A homogeneous system exists when both agents employ the same strategy or learning algorithm, with the same or similar parameters, which in our case are the discount factor (γ) and the memory type which could either be an LT or a feedforward NN. In contrast, a heterogeneous system exists when either both agents employ the

same algorithm with dissimilar parameters, or different algorithms. The opposing agents could be TD (Temporal Difference) [30], SARSA (State-Action-Reward-State-Action) [31] or Q-learning. The agents are provided only with incomplete information: they receive only the state of the environment, i.e., the actions of both the agent and the opponent in the previous round, but not the payoffs associated with each action. A Boltzmann exploration schedule is utilised, as it gives a good balance between exploration and exploitation. Specifically, an action a_i is selected from state s with probability $p(a_i)$ given by equation (1):

$$p(a_i) = \frac{e^{Q(s,a)/t}}{\sum_{a \in \{C,D\}} e^{Q(s,a)/t}} \quad (1)$$

where the temperature t is given by $t = 1 + 10 \times 0.995^n$, with n being the number of games played so far. The constants 1, 10 and 0.995 are chosen empirically. Each NN input is represented by 4 bits (thus 4 nodes), because according to the experiments in [29], this “unary encoding resulted in faster learning than a two-bit binary” one. Each bit is active depending on whether the previous action of the agent and the opponent was C or D. The network has 2 linear outputs corresponding to the estimates of the actions available in the game. One hidden layer with 3 sigmoidal activation function units is used and the network is trained by backpropagation as in [32], but with a single network.

2.2 Agents as Spiking Neural Networks

The game simulation is repeated with the two players implemented by two spiking NNs. The networks’ architecture is depicted in Fig. 1. Each network has two layers of leaky integrate-and-fire (LIF) neurons. The equation and values of the parameters used for modelling the LIF neurons are the same as in [23], apart from the value of the mean weight of the conductance used for the excitatory synapses which is set to 14nS. Both networks receive a common input of 60 Poisson spike trains grouped in four neural populations. The networks learn simultaneously but separately where each network seeks to maximise its own accumulated reward. Learning is implemented through reinforcement of stochastic synaptic transmission as in [23], where the model is developed along the hypothesis that microscopic randomness is harnessed by the brain for the purposes of learning. Briefly, within the model’s framework, each synapse acts as an agent pursuing reward maximisation through the actions of releasing or not a neurotransmitter upon arrival of a presynaptic spike. Each synapse records its recent actions through a dynamical variable, the eligibility trace [33], the time constant of which essentially reflects the way each synapse integrates time-related events. The input to the system is presented for 500ms and encodes the decisions the two networks had at the previous round. One can identify here a cyclic procedure which starts when the networks decide, continues by feeding this information to the networks during which learning takes place and ends by a new decision. Each network’s decision is encoded in the input, by the firing rate of two groups

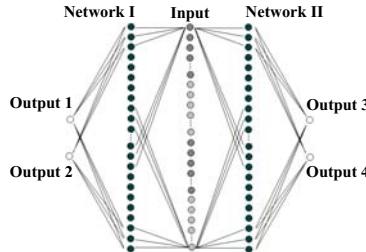


Fig. 1. Two spiking Neural Networks of Hedonistic Synapses and multilayer-type architecture compete in the IPD. Each network has two layers of hedonistic synapses that make full feedforward connections between three layers of neurons: the 60 shared input neurons, 60 leaky integrate-and-fire (LIF) hidden neurons and two LIF output neurons, randomly chosen to be either excitatory or inhibitory.

of Poisson spike trains. The first group will fire at 40Hz if the network cooperated and at 0Hz otherwise. The second group will fire at 40Hz if the network defected and at 0Hz otherwise. Consequently, four groups of Poisson spike trains provide the system's input with two groups always being active, preserving thus a balance at the output neurons' firing rates at the beginning of learning. Any significant difference in the output neurons' firing rate at any time should only be induced by learning and not by the differences of the driving input firing rates. At the end of each learning round the networks decide whether to cooperate or defect for the game's next round, according to the value each network assigns to the two actions. These values are reflected by the output neurons' firing rates at the end of each learning round. The cooperation value for network I and II is taken to be proportional to the firing rate of output neurons 1 and 3 respectively. Similarly, the defection value for network I and II is taken to be proportional to the firing rate of output neurons 2 and 4 respectively. When the two networks decide their play for the next round of the IPD, they each receive a distinct payoff given their actions and according to the game's payoff matrix (see Table II). This same payoff is also the global reinforcement signal (scaled down) that will train each network during the next learning round and thus guide the networks to their next decisions. Since the learning algorithm works with positive and negative reinforcements that are directly applied to the synapses and are extracted from the payoff matrix, it is then necessary that the payoff matrix contains both positive and negative values. Each network is reinforced for every spike of their output neuron that was "responsible" for the decision at the last round and hence for the payoff received. The networks thus learn through global reinforcement signals which strengthen the value of an action that elicited a reward and weaken the value of an action that resulted to a penalty.

In order to enhance competition between output neurons during a learning round, Seung's algorithm [23] is extended with additional global reinforcement signals administered to the networks for every output neuron spike not "responsible" for the decision. In the CD case for example, an additional reward of +1.15 is provided to network I for every spike of output neuron 2 and an additional

penalty of -1.15 is provided to network II for every spike of output neuron 3. The value of the action that was not chosen by each network is therefore also updated, by an opposite in sign reinforcement signal. The value of 1.15 applies to all outcomes and is chosen to be small enough such that: (i) any changes to the values of the players' actions are primarily induced by the reinforcement signals provided by the payoff matrix and (ii) not to cause IPD rules' violation.

Overall during a learning round, each network receives global, opposite in sign reinforcements for spikes of both of its output neurons. One of the two signals is due to the game's payoff matrix and its aim is to "encourage" or "discourage" the action that elicited reward or penalty and the other signal is complementary and its aim is to "encourage" or "discourage" the action that could have elicited reward or penalty if it had been chosen in the previous round of the game.

3 Results and Discussion

3.1 Lookup Table and Non-Spiking Neural Network Agents

The step-size parameter α , the backpropagation learning rate η and the discount factor γ were empirically set to 0.9 for all agents, as this showed good behaviour with fast convergence, especially when using LTs. Low α and η values, such as 0.1, prevented in some cases the algorithms from converging to the CC outcome. This may be due to the exploration schedule and more specifically to t (see eqn. 1) dropping at a level where no further exploration takes place. The games were run for 50 trials with 2000 rounds per trial.

As mentioned in Section 2.1, a multiagent learning system may either have homogeneous or heterogeneous settings. In homogeneous settings we test 3 Q-agents in self-play: 1) Q(LT, $\gamma=0.9$); 2) Q(LT, $\gamma=0.1$) and 3) Q(NN, $\gamma=0.9$). In heterogeneous settings we test 6 cases: 1) Q(LT, $\gamma=0.9$) vs TD(LT, $\gamma=0.9$) - different algorithm; 2) Q(LT, $\gamma=0.9$) vs SARSA(LT, $\gamma=0.9$) - different algorithm; 3) Q(LT, $\gamma=0.9$) vs Q(LT, $\gamma=0.1$) - different γ ; 4) Q(NN, $\gamma=0.9$) vs Q(LT, $\gamma=0.9$) - different memory type; 5) Q(NN, $\gamma=0.9$) vs TD(LT, $\gamma=0.9$) - different algorithm and memory type and 6) Q(NN, $\gamma=0.9$) vs SARSA(LT, $\gamma=0.9$) - different algorithm and memory type². Fig. 2 depicts the results taken when ranking all these cases based on the percentage of CC. As it can be seen, LTs with high γ obtain better results (93%) than (i) NNs with high γ (67%) and (ii) LTs with low γ (49%). This may be because (i) LTs are simpler and more suitable than NNs in our simple environment and (ii) a higher γ makes the agents take future rewards into account more strongly, which leads to an enhanced mutual cooperation. A successful heterogeneous case is the one where a Q agent competes with a SARSA, both with LTs and $\gamma=0.9$; in this case CC gets 86%. However, when the Q agent uses a NN, CC drops dramatically to 24% and DD

² It is worth noting that we could not evaluate SARSA and TD agents with NNs, nor Q with NN and a low γ ($=0.1$) or in some cases with many hidden nodes such as 10, because the algorithm diverges, since the NN weights are driven to infinity. Divergence problems are known to exist in RL with function approximators and some solutions were proposed to deal with them (see [34]; [35] and references therein).

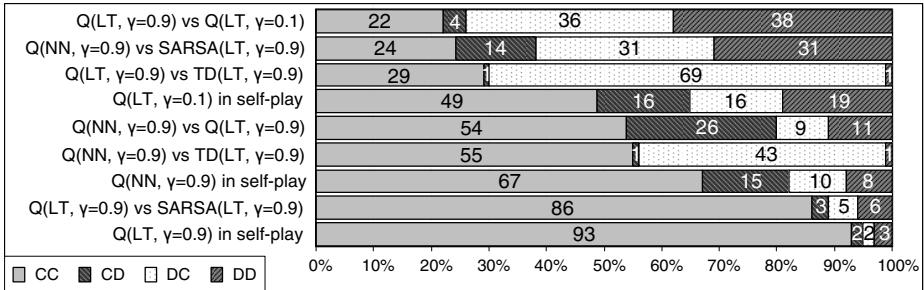


Fig. 2. Performance of Q vs Learning agents in homogeneous and heterogeneous environments ranked based on the CC percentage. Highest CC is achieved when an LT farsighted Q-agent (i.e., with a high discount factor γ) competes in self-play.

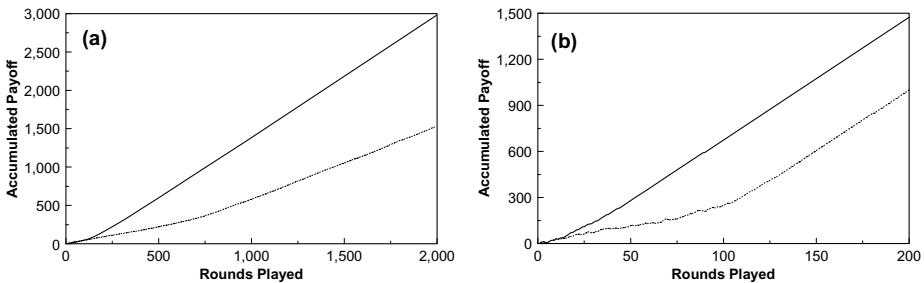


Fig. 3. Accumulated Payoff with: (a) non-spiking Q-agents (LTs) with the two players having a discount factor of 0.9 (solid line) and 0.1 (dotted line); (b) with spiking NNs with the two players having eligibility trace time constants 20ms (solid line) and 2ms (dotted line)

rises from 6% to 31%. The “strongest” algorithm in our case (i.e., Q-learning) accumulates more reward, since DC occurs in 31% of the rounds, whereas CD only in 14%. TD can be considered very weak when evaluated against Q-learning, as the latter manages to learn a better policy that exploits the weakness of its opponent. This is illustrated by the high DC percentages, in contrast with the low CD percentages. The interesting case of Q agents with LTs, with the row player being more farsighted than the column player, shows that being myopic does not mean that more unilateral defection would be played, as indicated by the percentages of DC (36%) and CD (4%). However, one may suggest that in this case a myopic agent in effect finds a worse policy, as it accumulates less payoff than the farsighted one. This is in line with the fact that in general, farsighted agents should accumulate more reward in the long-term than myopic ones. Finally, when an NN is compared with an LT, CC gets 54%. The reason for this could be in the difference in policy learning by the agents given that the LT learns faster than the NN agent and accumulates more reward (CD>DC). Fig. 3a illustrates the accumulated net payoffs over time for two homogeneous

cases, one with a farsighted and one with a myopic Q agent, both with LTs evaluated in self-play. It is clearly evident that in the case of the farsighted agent the accumulated payoff of the system is higher than the case of the myopic agent, which together with the results as presented in Fig. 2 support that CC is higher for farsighted agents.

3.2 Spiking Neural Network Agents

For the system configuration described in Section 2.2 a single game of the IPD consists of 200 rounds during which the two networks seek to maximise their individual accumulated payoff by cooperating or defecting at every round of the game. This simulation aims to investigate the capability of the spiking NNs to cooperate in the IPD. Two simulations were performed with the synapses of the two networks having different eligibility trace time constants which reflect how the networks integrate time related events and thus associated with the “memory” each network has. The values for both networks were set to 20ms and 2ms for the two simulations respectively. Therefore, during the first simulation the networks have a “good memory” whereas in the second one a “weak memory”. The results of both simulations are shown in Fig. 3b. The difference in the system’s performance is evident. When the system was configured with 20ms eligibility trace time constants, the accumulated payoff is much higher than the one with 2ms; this results from the difference in the cooperative outcome. With the eligibility trace time constants set at 20ms the two networks learned quickly to cooperate in order to maximise their long-term reward and achieved the CC outcome 182 out of the 200 times. On the contrary, when the system was configured with “weak memory”, learning took effect much later during the game (after the 100th round) and thus the system resulted in exhibiting much less cooperation (120 out of 200). However, the system with both configurations eventually managed to learn to cooperate. It is noted that the CC outcome not only persisted during the final rounds of the simulations, but it also did not change after a point (much earlier in the first case) due to the system’s dynamics that were evolved by that point in time in such a way to produce CC consistently. Results show that agents’ memory influences the cooperative outcome of the game in the sense that it delays it to a great extent. However, a weak memory does not destroy learning as the networks eventually learned to cooperate.

4 Conclusions

Our results indicate that the system accumulates higher cooperative reward when both agents have: (i) higher discount factor, in the case of non-spiking agents, or (ii) “stronger” memory, as in the case of the spiking agents with longer eligibility trace time constant. One may suggest that the effect of the discount factor in the non-spiking agents is equivalent to the effect of the “synaptic memory” of the spiking agents resulting from the use of eligibility traces, despite the fact that the former refers to future predictions, whereas the latter to past events. Based on this assumption, one could explain the more frequent emerging

of the cooperative outcome with high values of eligibility trace time constant and discount factor, in the spiking and non-spiking systems respectively. However, in order to make a direct comparison between the spiking and non-spiking systems we could either use eligibility traces in the Q-learning algorithms [28], or employ TD learning in our spiking NNs as in [36].

According to the results, apart from being desirable for the non-spiking agents to be farsighted in order to achieve the pareto-optimal outcome of the game (as mentioned above), they should also use LTs where possible (since LTs enable the system to converge faster and more accurately) and have learning algorithms of the same or of similar “strength” (such as both Q, or one Q and one SARSA). In the case of spiking agents, it has to be noted that our extension of the reinforcement of stochastic synaptic transmission of Seung [23], by enhancing competition between output neurons (see Section 2.2) through concurrently applying a positive global reinforcement to one output and a negative global reinforcement to the other output is both novel and necessary. More specifically it is essential, so as to avoid a positive feedback effect which would have increased the synaptic strength without bounds, leading to saturation of the synaptic connection and thus preventing further learning from taking place (like the limitation of classical Hebbian learning). Therefore one could conclude that in cases where more than one neuron competes for reinforcement in a spiking NN, the global evaluation signal of Seung’s reinforcement of stochastic synaptic transmission [23], should consist of global reward and penalty accordingly, for avoidance of possible synaptic saturation. In addition, it is also desirable for the payoff matrix for the non-spiking agents to mix positive and negative values (as in [37]), which if viewed as another technique of introducing competition into the system (as above), it could explain the enhancement of the cooperative outcome. As mentioned in Section 2.2, this mixture is necessary for the spiking agents.

In general, as it can be seen from the results, the behaviour of spiking and non-spiking agents is in effect similar (Fig. 3). We could therefore argue, that spiking agents could equally well be used in multiagent interactions as non-spiking agents. Certainly, a spiking agent system is more computationally expensive and should only be used when the task in question demands more biologically realistic models. For example, we have used a spiking multiagent system in modelling the high level behaviour of self-control [38].

Acknowledgments. We gratefully acknowledge the support of the University of Cyprus for a Small Size Internal Research Programme grant and the Cyprus Research Promotion Foundation as well as the European Union Structural Funds for grant PENEK/ENISX/0308/82.

References

1. Littman, M.L.: In: Cohen, W., Hirsh, H. (eds.) ICML, pp. 157–163. M Kaufmann, San Francisco (1994)
2. Hu, J., Wellman, M.P.: JMLR 4, 1039–1069 (2003)
3. Claus, C., Boutilier, C.: In: AAAI/IAAI, pp. 746–752. AAAI Press, Menlo Park (1998)

4. Littman, M.: In: Brodley, C., Danyluk, A. (eds.) ICML, pp. 322–328. M Kaufmann, San Francisco (2001)
5. Banerjee, B., Peng, J.: Convergent gradient ascent in general-sum games. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 1–9. Springer, Heidelberg (2002)
6. Greenwald, A.R., Hall, K.: In: Fawcett, T., Mishra, N. (eds.) ICML, pp. 242–249. AAAI Press, Menlo Park (2003)
7. Kapetanakis, S., Kudenko, D.: Reinforcement Learning of Coordination in Heterogeneous Cooperative Multi-agent Systems. In: Kudenko, D., Kazakov, D., Alonso, E. (eds.) AAMAS 2004. LNCS (LNAI), vol. 3394, pp. 119–131. Springer, Heidelberg (2005)
8. Bowling, M.H.: In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) NIPS, pp. 209–216 (2004)
9. Conitzer, V., Sandholm, T.: Mach. Learn. 67(1-2), 23–43 (2007)
10. Busoniu, L., Babuska, R., De Schutter, B.: IEEE Trans SMC C 38, 156–172 (2008)
11. Bowling, M., Veloso, M.: In: Nebel, B. (ed.) IJCAI, pp. 1021–1026. M Kaufmann, San Francisco (2001)
12. Powers, R., Shoham, Y., Vu, T.: Mach. Learn. 67(1-2), 45–76 (2007)
13. Shoham, Y., Powers, R., Grenager, T.: Artif. Intell. 171(7), 365–377 (2007)
14. Erev, I., Roth, A.E.: Artif. Intell. 171(7), 423–428 (2007)
15. Gordon, G.J.: Artif. Intell. 171(7), 392–401 (2007)
16. Fudenberg, D., Levine, D.K.: Artif. Intell. 171(7), 378–381 (2007)
17. Tuyls, K., Parsons, S.: Artif. Intell. 171(7), 406–416 (2007)
18. Stone, P.: Artif. Intell. 171(7), 402–405 (2007)
19. Xie, X., Seung, H.: Physical Review E 69(4), 41909 (2004)
20. Abbott, L., Nelson, S.: Nature Neuroscience 3, 1178–1183 (2000)
21. Florian, R.: Neural Comp. 19(6), 1468–1502 (2007)
22. Legenstein, R., Pecevski, D., Maass, W.: PLoS Comput Biol. 4(10), e1000180 (2008)
23. Seung, H.: Neuron 40(6), 1063–1073 (2003)
24. Rapoport, A., Chammah, A.: Prisoner’s dilemma: a study in conflict and cooperation. Univ. of Michigan Press, Ann Arbor (1965)
25. Nash, J.: PNAS 36(1), 48–49 (1950)
26. Fudenberg, D., Tirole, J.: Game Theory. MIT Press, Cambridge (1991)
27. Zinkevich, M., Greenwald, A., Littman, M.L.: Artif. Intell. 171(7), 440–447 (2007)
28. Watkins, C.: PhD thesis. Univ. of Cambridge (1989)
29. Sandholm, T., Crites, R.: Biosystems 37(1-2), 147–166 (1996)
30. Sutton, R.: Mach. Learn. 3(1), 9–44 (1988)
31. Rummery, G., Niranjan, M.: Technical Report CUED/F-INFENG/TR 166. Cambridge Univ. Eng. Dept. (1994)
32. Lin, L.J.: Mach. Learn. 8(3-4), 293–321 (1992)
33. Klopff, A.: The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence. Hemisphere Pub. (1982)
34. Baird, L., Moore, A.: In: Kearns, M., Solla, S., Cohn, D. (eds.) NIPS, pp. 968–974. MIT Press, Cambridge (1998)
35. Wiering, M.: Convergence and divergence in standard and averaging reinforcement learning. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 477–488. Springer, Heidelberg (2004)
36. Potjans, W., Morrison, A., Diesmann, M.: Neural Comp. 21(2), 301–339 (2009)
37. Kraines, D., Kraines, V.: In: Müller, J.P., Wooldridge, M., Jennings, N.R. (eds.) ECAI-WS 1996 and ATAL 1996. LNCS, vol. 1193, pp. 219–231. Springer, Heidelberg (1997)
38. Christodoulou, C., Banfield, G., Cleanthous, A.: J. Physiol., Paris (2009) (in press)

Unsupervised Learning in Reservoir Computing: Modeling Hippocampal Place Cells for Small Mobile Robots

Eric A. Antonelo^{*} and Benjamin Schrauwen

Electronics and Information Systems Department, Ghent University,
Sint Pietersnieuwstraat 41, 9000 Ghent, Belgium
`{eric.antonelo,benjamin.schrauwen}@elis.ugent.be`
<http://reslab.elis.ugent.be>

Abstract. Biological systems (e.g., rats) have efficient and robust localization abilities provided by the so called, place cells, which are found in the hippocampus of rodents and primates (these cells encode locations of the animal's environment). This work seeks to model these place cells by employing three (biologically plausible) techniques: Reservoir Computing (RC), Slow Feature Analysis (SFA), and Independent Component Analysis (ICA). The proposed architecture is composed of three layers, where the bottom layer is a dynamic reservoir of recurrent nodes with fixed weights. The upper layers (SFA and ICA) provides a self-organized formation of place cells, learned in an unsupervised way. Experiments show that a simulated mobile robot with 17 noisy short-range distance sensors is able to self-localize in its environment with the proposed architecture, forming a spatial representation which is dependent on the robot direction.

Keywords: Reservoir computing, slow feature analysis, place cells.

1 Introduction

Animals or robots should be able to efficiently self-localize in their environments for learning and accomplishing cognitive intelligent behaviors. They must be able to seek targets (energy resources) and accomplish important tasks in a efficient way. In this context, the ability to self-localize is clearly needed.

Standard robot localization systems are designed mostly by probabilistic methods which can perform SLAM (Simultaneous Localization And Mapping) under suitable assumptions [1] and are usually built for robots having high-resolution expensive laser scanners. Biologically inspired systems for robot localization can be considered a competitive alternative that works also for small mobile robots. Robustness, learning and low computation time are some characteristics of these biological inspired systems. Most systems are based on visual input from camera [2,3,4] and models hippocampal place cells from rats [2,3,4,5]. These place cells are the main components of the spatial navigation system in

^{*} This work was partially supported by FWO Flanders project G.0317.05.

rodents. Each place cell codes for a particular location of the rat's environment, presenting a peak response in the proximities of that location (the place field of that cell). Other components of the brain's spatial representation system includes head-direction cells, which encode the orientation of the animal in its environment, and grid cells, which are non-localized representations of space (having a grid-like structure of activations in space) [6]. Grid cells are found in the entorhinal cortex of rats and probably have an important role in the formation of place cells in the hippocampus [6]. Two classes of stimuli are available to place and grid cells: idiothetic and allothetic. Idiothetic input is originated from the physical body, such as proprioceptive sensors, which can be used for dead reckoning (path integration). Allothetic information is obtained from the external environment via sensors like distance sensors and camera. Dead reckoning can usually be corrected using allothetic information. The current work models place cells and to some extent, grid cells, endowing a simulated mobile robot with the capacity to self-localize in its environment through an unsupervised learning process. Reservoir Computing (RC) [7,8], Slow Feature Analysis (SFA) [9], and Independent Component Analysis (ICA) [10] are three (biologically plausible) techniques used in this work for modeling place cells. RC is a recently introduced paradigm in Recurrent Neural Networks (RNN) where the recurrent connections are not trained at all. Only output units are trained (usually in a supervised way) while the reservoir (the RNN itself) is a randomly generated dynamic system with fixed weights [8]. RC has biological foundations as it is shown, for example, that Liquid State Machines (a type of RC) are based on the micro-column structure in the cortex [11]. Furthermore, works such as in [12] establish a strong association between real brains and reservoirs. SFA is another recently proposed method to extract invariant or slowing varying features from input data [9]. Similarly to [3], we use SFA to model grid cells. While they use several SFA layers and high-dimensional input from a camera, we use only few noisy distance sensors and a RC-SFA based architecture.

This work proposes a general architecture based on reservoir computing and slow feature analysis (RC-SFA). While SFA provides an unsupervised learning mechanism for reservoirs, the latter provides short-term memory to SFA-based systems. This powerful combination can also be used in more general applications (such as speech recognition and robot behavior modeling). The proposed architecture is used here for autonomous map learning by modeling hippocampal place cells. A mobile robot with 17 short-range distance sensors is sufficient for generating a rather accurate spatial representation of maze-like environments without using proprioceptive information (odometry). The robot, in this way, autonomously learns to self-localize in its environment.

2 Methods

2.1 Reservoir Computing

The first layer of the RC-SFA architecture consists of a randomly created recurrent neural network, that is, the reservoir. This network is composed of sigmoidal neurons and is modeled by the following state update equation [8]:

$$\mathbf{x}(t+1) = f((1 - \alpha)\mathbf{x}(t) + \alpha(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}_{\text{res}}\mathbf{x}(t))), \quad (1)$$

where: $\mathbf{u}(t)$ denotes the input at time t ; $\mathbf{x}(t)$ represents the reservoir state; α is the leak rate [13]; and $f() = \tanh()$ is the hyperbolic tangent activation function (the most common type of activation function used for reservoirs). The connections between the nodes of the network are represented by weight matrices: \mathbf{W}_{in} is the connection matrix from input to reservoir and \mathbf{W}_{res} represents the recurrent connections between internal nodes. The initial state of the dynamical system is $\mathbf{x}(0) = \mathbf{0}$. A standard reservoir equation (without the leak rate) is found when $\alpha = 1$.

The matrices \mathbf{W}_{in} and \mathbf{W}_{res} are fixed and randomly created at the beginning. Each element of the connection matrix \mathbf{W}_{res} is drawn from a normal distribution with mean 0 and variance 1. The randomly created \mathbf{W}_{res} matrix is rescaled such that the system is stable and the reservoir has the echo state property (i.e., it has a fading memory [8]). This can be accomplished by rescaling the matrix so that the spectral radius $|\lambda_{\max}|$ (the largest absolute eigenvalue) of the linearized system is smaller than one [8]. Standard settings of $|\lambda_{\max}|$ lie in a range between 0.7 and 0.98 [8]. In this work we scale all reservoirs (\mathbf{W}_{res}) to a spectral radius of $|\lambda_{\max}| = 0.9$ which is an arbitrarily chosen value (shown to produce good results). The initialization of \mathbf{W}_{in} is given in Section 3.1.

The leak rate α should be in the interval $(0, 1]$ and can be used to tune the dynamics of the reservoir [13]. In this way, lower leak rates slow down the reservoir, increasing its memory but decreasing its capacity for agile processing of the input signal. Higher leak rates yield fast processing of the input but low memory to hold past stimuli. Similar results can be achieved when resampling the input signal for matching the timescale of the reservoir. For instance, it might be necessary to downsample an input signal if it varies too slowly. In this work, d_t represents the downsampling rate of the original input signal.

In this paper, the RC-SFA architecture is composed of a hierarchical network of nodes where the lower layer is the reservoir and the upper layers are composed of SFA and ICA units, respectively (Fig. 1). This hierarchical network learns in a unsupervised way (except for the reservoir whose weights (\mathbf{W}_{in} and \mathbf{W}_{res}) are kept fixed). The function of the reservoir is to map the inputs to a high-dimensional dynamic space. Because of its recurrent connections, the reservoir states contain *echoes* of the past inputs, providing a short-term memory to our model. The SFA layer receives signals from the input nodes $\mathbf{u}(t)$ and from the reservoir nodes $\mathbf{x}(t)$. This layer generates invariant or slowly varying signals [9] which are instantaneous functions of input from previous layers (see Section 2.2). The upper-most layer is composed of ICA units which generate a sparse and local representation of the slowing varying SFA features. The following sections focus on these upper layers. Next, consider n_u as the number of inputs; n_{res} as the number of neurons in the reservoir; n_{sfa} as the number of SFA units; and n_{ica} as the number of ICA units.

2.2 Slow Feature Analysis

Slow Feature Analysis (SFA) is a recently introduced algorithm that finds functions which are independent and slowly varying representations of the input [9].

SFA has also been shown to reproduce qualitative and quantitative properties of complex cells found in the primary visual cortex (V1) [14] and grid-cells from the entorhinal cortex of rats [3].

The learning task can be defined as follows. Given a high-dimensional input signal $\mathbf{x}(t)$, find a set of scalar functions $g_i(\mathbf{x}(t))$ so that the SFA output $y_i = g_i(\mathbf{x}(t))$ varies as slowly as possible and still carries significant information. In mathematical terms [9], find output signals $y_i = g_i(\mathbf{x}(t))$ such that:

$$\Delta(y_i) := \langle \dot{y}_i^2 \rangle_t \quad \text{is minimal} \quad (2)$$

under the constraints

$$\langle y_i \rangle_t = 0 \quad (\text{zero mean}) \quad (3)$$

$$\langle y_i^2 \rangle_t = 1 \quad (\text{unit variance}) \quad (4)$$

$$\forall j < i, \langle y_i y_j \rangle_t = 0 \quad (\text{decorrelation and order}) \quad (5)$$

where $\langle \cdot \rangle_t$ and \dot{y} denote temporal averaging and the derivative of y , respectively.

Learning: Before applying the algorithm, the input signal $\mathbf{x}(t)$ is normalized to have zero mean and unit variance. In this work, we only consider the linear case $g_i(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$, because the reservoir is already non-linear. The SFA algorithm is as follows:

Solve the generalized eigenvalue problem:

$$\mathbf{A}\mathbf{W} = \mathbf{B}\mathbf{W}\Lambda, \quad (6)$$

where $\mathbf{A} := \langle \dot{\mathbf{x}}\dot{\mathbf{x}}^T \rangle_t$ and $\mathbf{B} := \langle \mathbf{x}\mathbf{x}^T \rangle_t$.

The eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{n_{sfa}}$ corresponding to the ordered generalized eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{n_{sfa}}$ solve the learning task, satisfying (3-5) and minimizing (2) (see [9] for more details). This algorithm is guaranteed to find the global optimum.

Architecture: The SFA layer in our architecture (Fig. II) is denoted by $\mathbf{y}_{sfa}(t)$:

$$\mathbf{y}_{sfa}(t) = \mathbf{W}_{sfa} \mathbf{x}_{sfa}(t), \quad (7)$$

where: $\mathbf{x}_{sfa}(t)$ is the input vector at time t consisting of a concatenation of input $\mathbf{u}(t)$ and reservoir states $\mathbf{x}(t)$. Note that the states $\mathbf{x}(t)$ are generated by stimulating the reservoir with the input signal $\mathbf{u}(t)$ for $t = 1, 2, \dots, n_s$ by using (II), where n_s is the number of samples. The connection matrix \mathbf{W}_{sfa} is a $n_{sfa} \times (n_u + n_{res})$ matrix corresponding to the eigenvectors found by solving (6). In this work, the output signal $\mathbf{y}_{sfa}(t)$ generates non-localized representations of the environment, similarly to grid cells of the entorhinal cortex of rats [9].

2.3 Independent Component Analysis

Independent Component Analysis (ICA) is a method used for sparse coding of input data as well as for *blind source separation* [10]. The ICA model assumes that a linear mixture of signals x_1, x_2, \dots, x_n can be used for finding the n independent components or latent variables s_1, s_2, \dots, s_n . The observed values $\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_n(t)]$ can be written as:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) \quad (8)$$

where \mathbf{A} is the mixing matrix; and $\mathbf{s}(t) = [s_1(t), s_2(t) \dots s_n(t)]$ is the vector of independent components (both \mathbf{A} and $\mathbf{s}(t)$ are assumed to be unknown). The vector $\mathbf{s}(t)$ can be generated after estimating matrix \mathbf{A} :

$$\mathbf{s}(t) = \mathbf{W}\mathbf{x}(t) \quad (9)$$

where \mathbf{W} is the inverse matrix of \mathbf{A} . The basic assumption for ICA is that the components s_i are statistically independent. It is also assumed that the independent components have nongaussian distributions [10].

Learning: In this work the matrix \mathbf{W} is found with the FastICA algorithm [10]. Before using ICA, the observed vector $\mathbf{x}(t)$ is preprocessed by centering (zero-mean) and whitening (decorrelation and unit variance) [10]. FastICA uses a fixed-point iteration scheme for finding the maximum of the nongaussianity of $\mathbf{w}\mathbf{x}(t)$ (where \mathbf{w} is a weight vector of one neuron). The basic form of the FastICA algorithm (for one unit) is described next:

1. Initialize \mathbf{w} randomly
2. Let $\mathbf{w}^+ = E\{\mathbf{x}\mathbf{g}(\mathbf{w}^T\mathbf{x})\} - E\{\mathbf{g}'(\mathbf{w}^T\mathbf{x})\mathbf{w}\}$
3. Let $\mathbf{w} = \mathbf{w}^+/\|\mathbf{w}^+\|$
4. Do steps 2 and 3 until convergence,

where g is the derivative of a nonquadratic function G (in this work, $G(u) = u^3$) (see [10] for a detailed description).

Architecture: The equation for the ICA layer is (by redefining variables):

$$\mathbf{y}_{\text{ica}}(t) = \mathbf{W}_{\text{ica}}\mathbf{y}_{\text{sfa}}(t), \quad (10)$$

where: $\mathbf{y}_{\text{sfa}}(t)$ is the input vector at time t (the observed values); \mathbf{W}_{sfa} is the mixing matrix ($n_{\text{ica}} \times n_{\text{sfa}}$); and $\mathbf{y}_{\text{ica}}(t)$ is the output of the ICA layer (the independent components), which, in this work, learns to generate localized outputs which model hippocampal place cells of rats [6].

2.4 Robot Model

The robot model used in this work (Fig. 1) is part of the 2D SINAR simulator [15] and is described next. The robot interacts with the environment by distance and color sensors; and by one actuator which controls the movement direction (turning). Seventeen (17) sensor positions are distributed uniformly over the front of the robot (from -90° to $+90^\circ$). Each position holds two virtual sensors (for distance and color perception) [15]. The distance sensors are limited in range (i.e., they saturate for distances greater than 300 distance units (d.u.)) and are noisy (they exhibit Gaussian noise on their readings, generated from $N(0, 0.01)$). A value of 0 means near some object and a value of 1 means far or nothing detected. At each iteration the robot is able to execute a direction adjustment to the left or to the right in the range $[0, 15]$ degrees and the speed is constant (0.28 distance units (d.u.)/s). The SINAR controller (based on [15]) is an intelligent navigation system made of hierarchical neural networks which learn by interaction with the environment. After learning, the robot is able to efficiently navigate and explore environments, during which the signal $\mathbf{u}(t)$ is built by recording the 17 distance sensors of the robot.

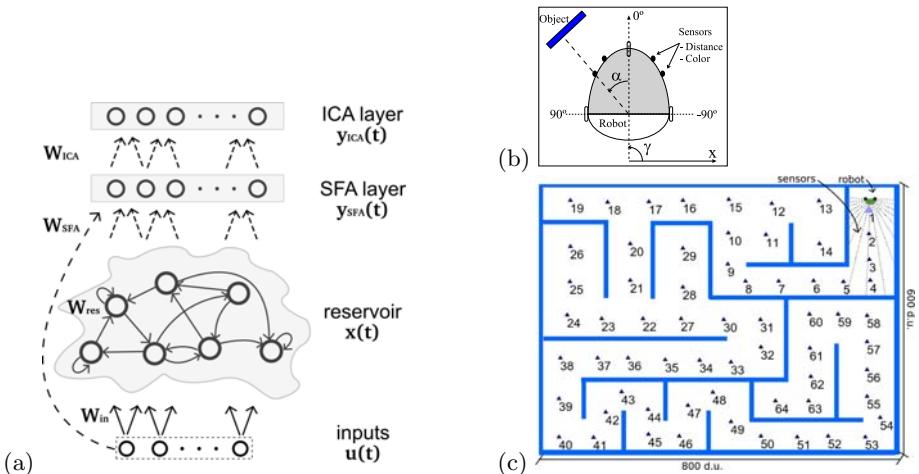


Fig. 1. (a) RC-SFA architecture. (b) Robot model. (c) Environment E1. The environment is tagged with 64 labels displayed by small triangles.

3 Experiments

3.1 Introduction

In the following, we describe the architecture (Fig. 1) and the initialization of parameters for the experiments. The first layer of the architecture corresponds to a dynamic reservoir of 400 neurons, which provides short-term memory to our model. The second layer consists of 70 SFA units, which extracts the slow features from the reservoir states and distance sensors. The output of the SFA layer models the grid cells found in the entorhinal cortex of primates [6], similarly to the simulated rat's grid cells formed from visual input in [3]. The last layer is composed of 70 ICA units, which model place cells usually found in the CA areas of the hippocampus [6]. Grid cells are non-localized in the sense that they fire for more than a single location while place cells encode a specific position of the animal's environment.

As the robot has a very low speed, the input signal (17 distance sensors) is downsampled by a factor of $d_t = 50$ (using the matlab function *resample*). Additionally, the leak rate in the reservoir is set to $\alpha = 0.6$. These settings, optimized for the supervised scheme in [16], worked well for the current work (optimization and performance analysis of these parameters is left as future work). The matrix connecting the input to the reservoir (W_{in}) is initialized to -0.2, 0.2 and 0 with probabilities 0.15, 0.15 and 0.7, respectively.

The experiments are conducted using environment E1 (Fig. 1). It is a big maze with 64 predefined locations spread evenly around the environment (represented by small labeled triangles). First, for generating the input signal, the simulated robot navigates in the environment for 350.000 timesteps while its distance sensor measurements are recorded (the robot takes approximately 13.000 timesteps to visit most of the locations). The controller basically makes the robot explore

the whole environment. After dowsampling the recorded input signal $\mathbf{u}(t)$, the number of samples becomes $n_s = 7.000$. Next, the downsampled input signal is used to generate the reservoir states $\mathbf{x}(t), t = 1, 2, \dots, n_s$ using (1).

The learning of the RC-SFA architecture takes place in 2 steps and uses 5/6 of the input signal as the training dataset (1/6 for testing). First, the SFA layer learns by solving (6) where the inputs are the reservoir states and distance sensors (like in (7)). After \mathbf{W}_{sfa} is found, the output of SFA units $\mathbf{y}_{\text{sfa}}(t), t = 1, 2, \dots, n_s$ is generated using (7). The second step corresponds to the learning of the upper ICA layer by applying the FastICA algorithm from Section 2.3 where the inputs for this layer are the output of the SFA units. The output signals $\mathbf{y}_{\text{sfa}}(t)$ and $\mathbf{y}_{\text{ica}}(t)$ are upsampled to the original sampling rate of $\mathbf{u}(t)$.

3.2 Results

The RC-SFA architecture is trained sequentially from the middle SFA layer to the top ICA layer. This section shows the results after training the layers with the input signal $\mathbf{u}(t)$ and reservoir states $\mathbf{x}(t)$ generated with the previously presented setup. Fig. 2(a) shows the output of 3 SFA units for a test input signal. The left plots show the outputs over time whereas the right plots show the response of the neurons as a function of the robot position in the environment. In the left plot, the horizontal axis represents the time, the left vertical axis denotes the real robot location (as given by the labeled triangles in Fig. 1), and the right vertical axis denotes the SFA output of the neuron. The colored dots represent the output of the SFA unit (where red denotes a peak response, green an intermediate response, and blue a low response). The SFA output is also shown as a black line in the same plot and as a colored trajectory in the right plot. As SFA units are ordered by slowness, the first SFA unit has the slowest response. It is high only for two areas of the environment: locations 10 to 17, and locations 27 to 35. Units 12 and 24 vary much faster, encoding several locations of the environment. In the same figure, it is possible to observe that SFA units learn a representation which is dependent on the robot heading. For instance, unit 12 responds differently for trajectories which go towards location 64 and trajectories that start from this position. As slow feature analysis is a method which is highly dependent on the input statistics, the movement patterns generated by the robot controller decisively influence the learning of the SFA units. In this context, most SFA units learn to be robot direction dependent in the current experiment. However, if the robot would change its direction more often (or faster compared to its speed), the SFA unit could learn to be direction invariant (by learning the slow features, that is, the robot position).

The upper ICA layer builds on the SFA layer. During learning, ICA units seek to maximize nongaussianity so that their responses become sparse and clustered and also as independent as possible. This form of sparse coding lead to the unsupervised formation of place cells. Fig. 2(b) shows a number of ICA units which code for specific locations in the environment. These units were chosen such that they code for adjacent locations as if the robot was navigating in the environment. The peak response is represented by white dots while lower

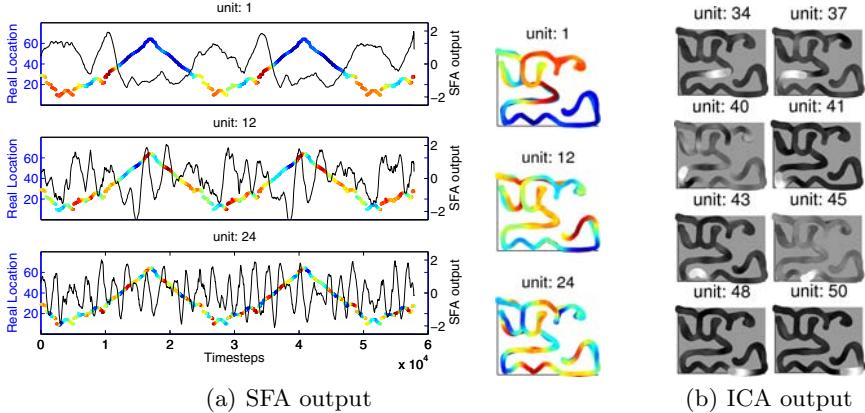


Fig. 2. Results for simulations in Environment E1. (a) Responses of SFA units 1, 12, and 24. Left: the SFA output over time. For each location (in time) given by the labeled triangles in Fig. II, there is a colored dot where red denotes a peak response, green an intermediate response, and blue a low response. The output is also plotted as a black line. Right: the same SFA output as a function of the robot position. (b) Response of ICA units as a function of the robot position. White dots denote high activity while darker dots represent lower responses. The results show the localized aspect of place cells or ICA units (the peak response is characteristic of one specific location).

responses are given gradually in darker colors. In order to view the localized aspect of place cells more clearly, the output of ICA units are ordered such that they have a spatial relationship. The reference locations (from 1 to 64), shown in environment E1 (Fig. II), are used to automatically order the ICA layer. ICA units which do not respond strongly enough (that is, less than 4.5) in any situation are set to the end of the vector. Fig. 3(a) shows the real occupancy grid for the robot while it drives in environment E1 and the respective ICA activation map showing the spatially-ordered ICA responses (where $u(t)$ is a test signal not used during learning). The peak responses are shown in black while white dots represent lower responses. Eleven ICA units (from 59 to 70) did not fire strongly enough and, so, did not code for any location. This activation map is very similar to the real robot occupancy grid showing that the place cells efficiently mapped most of the environment. Fig. 3(b) shows a magnification of the ICA activation map for locations under 20. It is possible to note that for almost the whole time period there is only a single ICA unit active (i.e., the ICA layer is detecting locations most of the time). This figure also clearly shows that most ICA units are dependent on the robot direction (as SFA units are). We have repeated the experiments shown here with the same datasets more than 15 times (where for each time a different random reservoir is created) with no visible changes in the learned place cells. Furthermore, preliminary results show that the RC-SFA architecture also works for other kinds of environment configurations and environments with dynamic objects.

The importance of the reservoir can become more evident as it provides a short-term memory of previous inputs. In order to compare results, the proposed

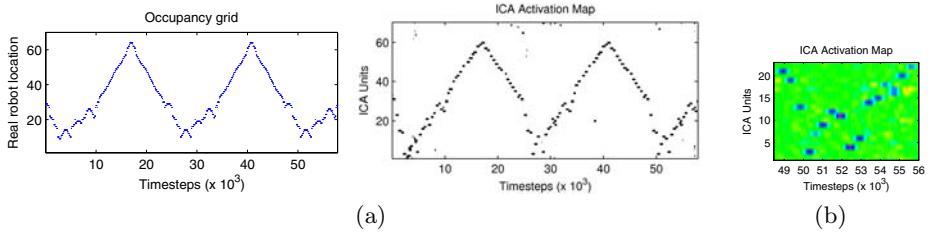


Fig. 3. Emergence of place cells in environment E1. (a) The real robot occupancy grid (left) and the respective spatially-ordered ICA activation map (black dots denote peak responses and white represent lower responses). (b) Close view of the ICA activation map where green denotes low response and blue high response.

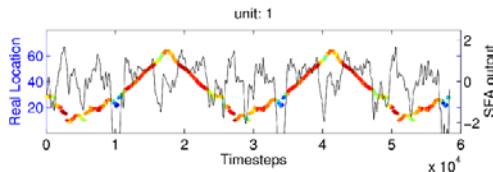


Fig. 4. Results for modified architecture without the reservoir layer in environment E1: the slowest SFA unit output over time, as in Fig. 2

architecture in Fig. 1 is modified so that the inputs connect directly to the SFA layer and no reservoir is used at all. The following changes are also accomplished: the downsampling rate is increased to $d_t = 200$ for slowing down the input signal; $n_{sfa} = 100$ and $n_{ica} = 100$. The SFA algorithm for this experiment includes a quadratic expansion process on the inputs [9], making the SFA effectively a non-linear process. The slowest SFA unit is shown in Fig. 4. The response pattern from this unit seems more noisy than the case when using the RC-SFA architecture which shows a smooth signal (Fig. 2(a)). The ICA activation map for this architecture (not shown) was very fuzzy and far from the one obtained in Fig. 3 (also did not solve the perceptual aliasing problem once one ICA unit coded for multiple similar but distinct locations). In this way, this modified setup without the reservoir was not able to model place cells in the current experiment.

4 Conclusion and Future Work

This work proposes a new biologically inspired architecture (RC-SFA) based on a mixture of recently developed techniques (Reservoir computing, Slow Feature Analysis and Independent Component Analysis). The RC-SFA is a general architecture which can be easily applied to wide range of applications (e.g., extract slowly-varying components of an input signal such as: behaviors or movements of a humanoid robot or phonemes and words from speech data). In this work, a simulated mobile robot (with few proximity sensors) autonomously learns, using our proposed RC-SFA architecture, a rather accurate spatial representation of its environment and, in this way, to self-localize in it. We do not use proprioceptive

information as a form of spatial memory, but rather the short-term memory of the reservoir has shown to eliminate the perceptual aliasing from sensors, producing a system which autonomously learns the relevant information from the input stream. Further interesting directions for research include the validation of the proposed architecture with a real robot (such as the e-puck robot) and accomplish further experiments such as kidnapping the robot and navigation in a dynamic environment. Other important questions include: how distinct settings of reservoir parameters influence the learning of the spatial representation; what kind of environments can be used; and what range of noise and sensor failures the model can cope with.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. The MIT Press, Cambridge (2005)
2. Arleo, A., Smeraldi, F., Gerstner, W.: Cognitive navigation based on nonuniform gabor space sampling, unsupervised growing networks, and reinforcement learning. *IEEE Transactions on Neural Networks* 15(3), 639–652 (2004)
3. Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Comput. Biol.* 3(8), 1605–1622 (2007)
4. Stroesslin, T., Sheynikhovich, D., Chavarriaga, R., Gerstner, W.: Robust self-localisation and navigation based on hippocampal place cells. *Neural Networks* 18(9), 1125–1140 (2005)
5. Chavarriaga, R., Strsslin, T., Sheynikhovich, D., Gerstner, W.: A computational model of parallel navigation systems in rodents. *Neuroinformatics* 3, 223–241 (2005)
6. Moser, E.I., Kropff, E., Moser, M.B.: Place cells, grid cells and the brains spatial representation system. *Annual Reviews of Neuroscience* 31, 69–89 (2008)
7. Schrauwen, B., Verstraeten, D., Van Campenhout, J.: An overview of reservoir computing: theory, applications and implementations. In: Proc. of ESANN (2007)
8. Jaeger, H.: The “echo state” approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001)
9. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: Unsupervised learning of invariances. *Neural Computation* 14(4), 715–770 (2002)
10. Hyvärinen, A., Oja, E.: Independent component analysis: algorithms and applications. *Neural Networks* 13, 411–430 (2000)
11. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14(11), 2531–2560 (2002)
12. Yamazaki, T., Tanaka, S.: The cerebellum as a liquid state machine. *Neural Networks* 20, 290–297 (2007)
13. Jaeger, H., Lukosevicius, M., Popovici, D.: Optimization and applications of echo state networks with leaky integrator neurons. *Neural Networks* 20, 335–352 (2007)
14. Berkes, P., Wiskott, L.: Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision* 5, 579–602 (2005)
15. Antonelo, E.A., Baerlvedt, A.J., Rognvaldsson, T., Figueiredo, M.: Modular neural network and classical reinforcement learning for autonomous robot navigation: Inhibiting undesirable behaviors. In: Proceedings of IJCNN, Vancouver, Canada, pp. 498–505 (2006)
16. Antonelo, E.A., Schrauwen, B., Stroobandt, D.: Event detection and localization for small mobile robots using reservoir computing. *Neural Networks* 21, 862–871 (2008)

Switching Hidden Markov Models for Learning of Motion Patterns in Videos*

Matthias Höffken**, Daniel Oberhoff, and Marina Kolesnik

Fraunhofer Institut FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
matthias.hoeffken@fit.fraunhofer.de

Abstract. Building on the current understanding of neural architecture of the visual cortex, we present a graphical model for learning and classification of motion patterns in videos. The model is composed of an arbitrary amount of Hidden Markov Models (HMMs) with shared Gaussian mixture models. The novel extension of our model is the use of additional Markov chain, serving as a switch for indicating the currently active HMM. We therefore call the model a Switching Hidden Markov Model (SHMM). SHMM learns from input optical flow in an unsupervised fashion. Functionality of the model is tested with artificially simulated time sequences. Tests with real videos show that the model is capable of learning and recognition of motion activities of single individuals, and for classification of motion patterns exhibited by groups of people. Classification rates of about 75 percent for real videos are satisfactory taking into account a relative simplicity of the model.

1 Introduction

As the amount of acquired, stored, and transmitted video data increases every day, the need for automatic comprehension and sorting of this data becomes ever more crucial. Having a machine performing pre-selection of video data upon a given criterion, would diminish the amount of data to be stored for human visual analysis. This is particularly important for all kinds of video surveillance, in which human operators are routinely stressed by a huge inflow of video data. An automatic system could either issue a warning signal for operator if a given criterion is met in the visual input, or the whole video record could be automatically annotated to simplify its visual screening later on.

One promising approach to solve the above problem is to design a system capable of learning object appearances and motion patterns of events from the video input. A number of efforts have been undertaken to model appearance based action recognition either as competitive analysis tools ([1],[2],[3]), or by modelling available psychophysical data [4].

* The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n°215866, project SEARISE.

** Corresponding author.

The motivation in this work is to utilise knowledge from neurophysiological and psychophysical research in order to build computer systems for automatic motion analysis. We do this by transforming major findings from such research into mathematically sound statistical models. The systems we design consist of a set of relatively small components, which may be seen as abstractions of brain areas or parts thereof, and mechanisms linking these components into larger and much more powerful systems for visual analysis. Since humans and animals are capable of learning autonomously, these components and the combined systems replicate this capability. We thus formulate our systems in terms of generative Bayesian statistical models in contrast to purely discriminative approaches.

While recognising the extensive work done in this area, we nevertheless see a strong need for autonomously learning systems, as the largest part of those approaches includes either strong supervision or only weak learning strategies. One promising example devised in the group of Poggio et. al. [3], features a neurophysiologically inspired system that, like its predecessor HMAX, does not contain any learning algorithm beyond random learning. Even though this system only uses supervised feature selection, it shows how much a system can gain by just roughly following important neurophysiological principles. The work by Giese et. al. [4] demonstrates how a system that is in many ways similar to ours, can explain the existing psychophysical data. The group of Dean et. al. [5] have done a groundwork on hierarchical graphical models for computer vision. Their work provides great inspiration and technical background, but, to our knowledge, it neither been applied to real video data, nor it has been tried to perform sequence learning.

The most advanced model comparable to the one presented here, has been constructed in the group of Hinton et. al. [6]. The drawback of their approach is the use of gradient descent learning, which typically requires thousands of iterations over the data to converge. The latter seems prohibitive for large video sequences. Several other groups (c.f. [7], [8]) have experimented with switching sequence models comparable to ours, but none of them seems to have considered a hierarchical approach, nor do any of them tried an unsupervised learning.

2 Model

Input to our model is the optical flow computed by a neurally inspired algorithm for recurrent motion estimation suggested in [9] and further extended in [10], [11]. Locally estimated motion hypothesis are then used for evaluation of an expected motion vector at each pixel location.

Our model consists of four consecutive feed-forward steps. Each step performs one of the following processing operations (Fig. 1a). First, receptive fields are pooled within a local spatial (and/or temporal) neighbourhood. The pooling decreases spatial resolution of resulting feature vectors while preserving a spatio-temporal mapping of their receptive fields. A pooling region is selected manually depending on the scale of the scenario (see Sect. 3) and is performed via averaging. Second step performs Hotelling transform of the feature vectors

in order to decrease the number of variables to be learnt. Third, vector quantisation (clustering) [12] using Gaussian generative mixture model is carried out. Finally, the fourth step uses the obtained mixture components as an emission model for the learning of sequences of motion patterns as follows.

Time dependencies between motion patterns are modelled by SHMM (Fig. 1b) consisting of $R+1$ Markov chains. R *sub-chains* are emitting; they have an equal number of internal states and share the same emission model. At any given time moment, only one chain out of these R emitting sub-chains is active and emits a data vector, while the others are proceeding without emissions. Hidden states of the inactive sub-chains are propagated independently according to their transition matrices. In fact, this corresponds to a mixture of R Hidden Markov Models (HMM).

The $R+1$ chain, referred as *switch-chain*, has R possible hidden states. These hidden states indicate the active emitting sub-chain at each given moment. This chain actually models the time dependency between the different subsequences of motion patterns, which are modelled by the sub-chains. Specifically, each diagonal element of the switch chain transition matrix determines how long the corresponding sub-chain is expected to be active.

The full joint probability, $P(\{Y, X, S\})$, of the model is given by:

$$\begin{aligned} P(\{Y, X, S\}) = & P(S^{(1)}) \prod_{t=2}^T P(S^{(t)} | S^{(t-1)}, \Theta_S) \cdot P(\Theta_S | \alpha, \alpha_0) \\ & \cdot \prod_{r=1}^R \left[P(X_r^{(1)}) \prod_{t=2}^T P(X_r^{(t)} | X_r^{(t-1)}, \Theta_{X_r}) \right] \\ & \cdot \prod_{t=1}^T P(Y^{(t)} | X_1^{(t)}, \dots, X_R^{(t)}, S^{(t)}) \end{aligned} \quad (1)$$

where $Y^{(t)}$ is the emission of the model at the moment, t , resulted from the Hotelling transform, $X_r^{(t)}$ indicates the hidden state of the r -th sub-chain, $S^{(t)}$ represents the hidden state of the switch-chain, T is the observation time, and Θ_S, Θ_{X_r} are the parameters (transition matrices) of the respective Markov chains. $P(\Theta_S | \alpha, \alpha_0)$ is a Dirichlet-Prior that allows to regularise the learning of the switch-chain parameters during the EM-Algorithm. We use $\alpha \in [0, 1]$ as the expectation value for the diagonal of the switch transition matrix. The expectation value for the non-diagonal elements is then $\beta = (1 - \alpha)/(R - 1)$. α_0 indicates the impact of the prior during the M-Step and should be adapted to the number of data and classes. Hence the parameters for applied Dirichlet distribution are then $\alpha_0 \cdot \alpha$ for the diagonal and $\alpha_0 \cdot \beta$ for non diagonal values. The regularisation can be deactivated by setting $\alpha = 1/R$ and $\alpha_0 = R$.

2.1 Learning and Classification

Except for the first pooling step, whose parameters are kept fixed, parameters in steps 2 through 4 are learnt in a batch procedure.

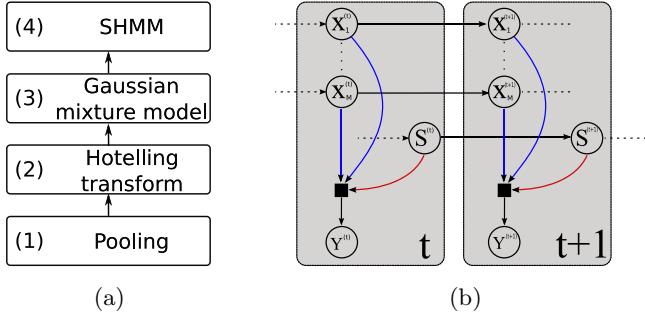


Fig. 1. (a) Model's four consecutive feed-forward steps. (b) Temporal dependencies between two consecutive time steps in SHMM. X_m denote hidden states of sub-chains, S is the switch state selecting the currently active sub-chain.

We use the Expectation Maximisation (EM) algorithm for the learning of the mixture model at step 3 and the sequence model at step 4.

The Expectation step for the learning of all $R+1$ Markov chains is similar to the forward-backward algorithm applied for the learning of HMM's [13], but differs in two important points. First, performing the forward as well as the backward step requires storing all hypothesis of all intermediate hidden variables. In the context of video processing and applications requiring on-line learning, this is prohibitively expensive and therefore only the forward step is performed. Second, the probability of a sub-chain to emit a feature vector depends on the state of the last time step and the probability of it being active. The latter probability is received from the switch-chain. The switch-chain takes a decision on the active sub-chain based on its own previous internal state and the different emission probabilities of all sub-chains. These dependencies can be solved as follows. The expectation value of the hidden switch state, $S_r^{(t)}$, is given by:

$$\langle S_r^{(t)} \rangle = \frac{1}{C_1} \cdot \left[\sum_{\forall m} P(Y^{(t)} | X_{rm}^{(t)}) \sum_{\forall m'} \langle X_{rm'}^{(t-1)} \rangle P(X_{rm}^{(t)} | X_{rm'}^{(t-1)}) \right] \\ \cdot \left[\sum_{\forall r'} \langle S_{r'}^{(t-1)} \rangle P(S_r^{(t)} | S_{r'}^{(t-1)}) \right] \quad (2)$$

and the expectation value of the hidden sub-chain states, $X_{rm}^{(t)}$, is given by:

$$\langle X_{rm}^{(t)} \rangle = \frac{1}{C_2} \cdot \left[\langle S_r^{(t)} \rangle P(Y^{(t)} | X_{rm}^{(t)}) + (1 - \langle S_r^{(t)} \rangle) \right] \quad (3)$$

$$\cdot \sum_{\forall m'} \langle X_{rm'}^{(t-1)} \rangle P(X_{rm}^{(t)} | X_{rm'}^{(t-1)}) \quad (4)$$

here C_1 and C_2 are normalising factors, indexes r, r' reffer to sub-chains, and m, m' denote emission states.

Given the hypothesis about all hidden model states, the parameter estimates that maximise the lower bound of the model likelihood in Eq. 1 can be obtained

by differentiating the log-likelihood subject to the constraint on the initial and transition probabilities summing to one. The derivation of the corresponding parameter estimates is slightly more complex than in case of ordinary HMMs.

Time complexity of E- and M-Steps for both switch-chain and sub-chains is equal to the time complexity of a single HMM. Hence, M emission states would require $O(T(RM^2 + R^2))$ operations.

For classification we either directly employ the $S^{(t)}$ -distribution, obtained from the E-Step or feed it in a naïve Bayesian classifier. While in the first case one sub-chain is trained for each class, second case allows using multiple sub-chains.

3 Experiments

Being a generative model, the SHMM is capable of unsupervised learning, which makes it suitable for construction of large hierarchical systems. This enables a wide range of possible applications. To evaluate the model performance, we connect its output with a naïve Bayesian classifier.

We test the performance of the SHMM learning in three different sequences of experiments. These experiments, however, do not aim at outperforming established and task-tuned supervised classifications methods, rather they should be seen as a proof of concept of our generic approach to unsupervised hierarchical learning.

3.1 Bootstrap

The functionality of the SHMM implementation was first verified in several bootstrap tests. For these tests we sample an artificial data set, consisting of the active states, $S^{(t)}$, and the corresponding hidden sub-chain states, $X_r^{(t)}$, as generated by a simple SHMM with known parameters. The sampled data set is used for the training of SHMM and for the validation of its classification rates as described in Sect. 2.1. This SHMM (Fig. 2a) consists of two sub-chains with five states and a switch-chain with a high probability of 0.95 to remain in the same active state. The hidden sub-chain states 1 and 2 are preferred by the first sub-chain whereas second sub-chain prefers the states 4 and 5. State 3 is shared by both sub-chains. We test both supervised ($S^{(t)}$ is part of the training data set) and unsupervised (only $X^{(t)}$ is known) learning. Furthermore, we vary the training procedure by, first, learning the sub-chains together with the switch-chain, and, second, by learning only the sub-chains.

SHMM classification performance reaches about 85% correct classifications for unsupervised learning (Fig. 2b) and 97% for supervised learning. Our experiments further show that the learning of the sub-chain transition matrices together with the switch-chain transition matrix may become unstable in some cases. This behaviour is mainly influenced by the initialisation of the switch-chain transition matrix before applying the EM-algorithm. In case of an initial transition matrix with high values on its diagonal, only one sub-chain is trained and the switch transition matrix becomes a unity matrix after a few iterations. In the opposite case of an initial transition matrix with small diagonal values, the active state often switches between the sub-chains thus hindering the sub-chains from learning different sequence patterns. This can be overcome by either

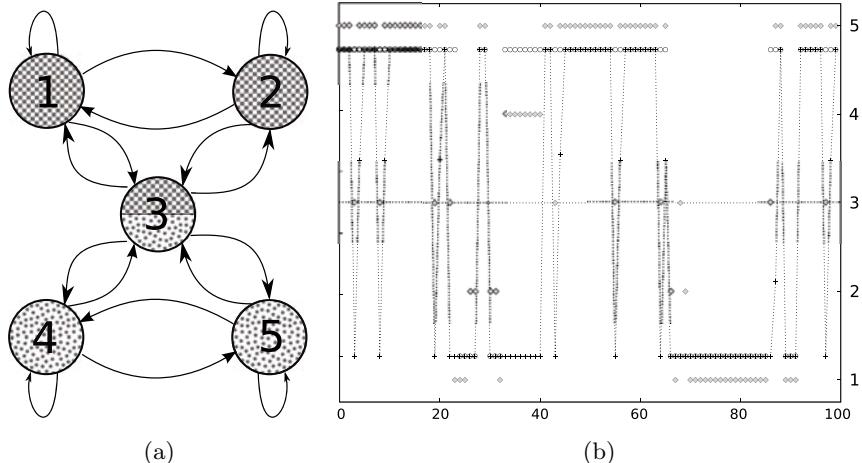


Fig. 2. [a] SHMM employed in the bootstrap tests. Two different sub-chains are shown in checked and dotted patterns with arrows indicating transitions with probabilities higher than 0.1. Both training and test data have been sampled by this SHMM. [b] Classification results over a sequence of 100 time steps sampled by SHMM depicted in [a]. Circles denote the correct switch-states $S_0^{(t)}$. Crosses indicate their expectation values $\langle S_0^{(t)} \rangle$ as estimated during the classification. Y-axis on the left side plots the expectation values. Gray diamonds show the sampled sub-chain states $X^{(t)}$, plotted on Y-axis to the right. Training in [b] was done in unsupervised way.

excluding the switch-chain from the learning procedure or by regularizing the switch transition matrix towards more reasonable values by using the Dirichlet prior $P(\Theta_S | \alpha, \alpha_0)$ during the EM-training. For the latter solution we reach best results for priors promoting diagonal switch chain transition values of 0.7.

3.2 Single Human Action

For this experiment we use the public Weizmann human action dataset¹ of 90 video sequences and compare the results with those presented in [14]. The video sequences show nine different people, each one performing 10 natural actions such as “run,” “walk”, “skip”, “jumping-jack” (or shortly “jack”), “jump-forward-on-two-legs” (or “jump”), “jump-in-place-on-two-legs” (or “pjump”), “gallop-sideways” (or “side”), “wave-two-hands” (or “wave2”), “wave-one-hand” (or “wave1”), or “bend” (Fig. 3). The video sequences are also complemented by the corresponding silhouettes of acting people segmented from background in each video frame using background subtraction. Except for “wave1”, “wave2” and “bend” all classes contain videos where persons perform actions in both directions, left to right and right to left. Since our model is not direction-invariant we synchronize the moving directions by vertically mirroring the videos showing action from right to left.

¹ Downloaded at

<http://www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html>

In the experiment we generate a bounding box with fixed size of 96x64 pixels covering the largest silhouette segment in all 90 video sequences and centring them at the centre of gravity for a given frame. Optical flow values assigned to pixels within these bounding boxes are used for the learning and classification of motion patterns.

During the first processing step applied to each frame, velocity vectors from patches of size 16x16 pixels within bounding boxes are pooled using a weighted average, with weights equal to the norm of the corresponding velocity vectors. Next, feature vectors are computed by collecting weighted averages over patches of 6x4 pixels and concatenating them into long vectors.

The Gaussian mixture model comprises 30 mixture components for the clustering step and the SHMM with 20 sub-chains learns sequence patterns in a frame-wise way. For the prior parameters we used $\alpha = 0.7$ and $\alpha_0 = 200$, corresponding to approximately 50% of the expected number of data assigned to each sub-chain. For classification we use the $\langle S_r^{(t)} \rangle$ estimates obtained from step 4 (Sect. 2.1) to train a naïve Bayesian Classifier with 10 different action classes.

Similarly to [14] we use 9 video sequences for training leaving the last one for classification. We permute videos in the leave-one-out procedure obtaining 10 independent experiments. The overall classification rate in this experiment averages to 76.3% and the confusion Table 1 illustrates the classification results.

Clearly, we do not reach the high classification rates reported in [14] where a supervised learning/classification method makes use of 3-dimensional shapes, which are obtained from stacks of video silhouettes. One reason for this is that in contrast to [14], the SHMM uses unsupervised learning. Next, SHMM uses first-order Markov models in time, whereas [14] exploits a much higher order temporal information collected over eight frames by constructing space-time cubes from pre-segmented silhouettes. Besides, our model only uses bounding box segmentation of each frame.

Another reason is the option of switching between states of different HMMs during the learning. For instance in ambiguous case with frames showing motion patterns that occur in several (or all) action classes, the model has the unfavourable possibility of switching to a wrong sub-chain. Such ambiguous frames typically appear in the "bend", "wave1" and "wave2" classes, when acting person stands still for a short while. On the other hand this switching capability would allow learning different subsequent actions in a single video sequence.

3.3 Group Action

In this experiment, we test the SHMM model on a manually annotated video showing a spectator tribune during a soccer game. The video exhibits four different types of spectator's behaviour labelled as (Fig. 4a) : FLAG (showing waiving flag), JUMP (jumping group of soccer fans), GROUP (group of people with few motion activities) and BGROUND (remaining class with no predominant activity) (depicted in Fig. 4a).

The feature vectors in the labelled regions are constructed from the optical flow by pooling 8x8 pixel patches using a weighted average followed by the

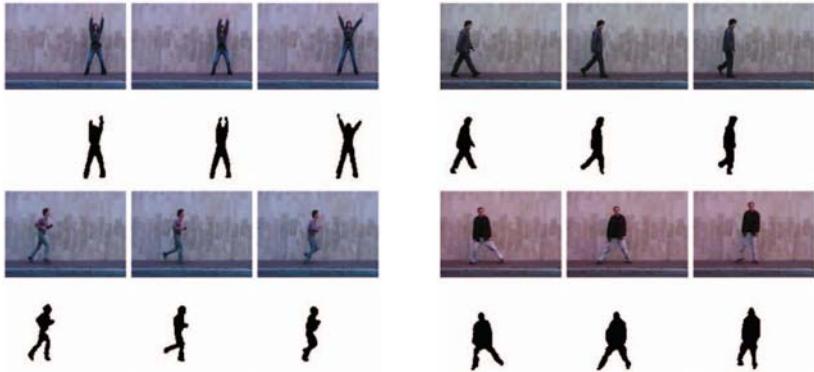


Fig. 3. Examples of video sequences and corresponding silhouettes from the Weizmann human action dataset [14]

Table 1. Confusion matrix in the classification experiment with 10 different actions. Classification of the SHMM states is done by the naïve Bayesian Classifier. Empty entries are zero. a1=“walk”, a2=“run”, a3=“skip”, a4=“jack”, a5=“jump”, a6=“pjump”, a7=“side”, a8=“wave1”, a9=“wave2”, a10=“bend”.

	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10
a1	69.2	3.2	6.8		17.4		3.5			
a2	2.5	68.6	22.2		3.9		1.5			1.2
a3	4.3	9.6	75.3		8.1		2.8			
a4				92.4		4.9		0.3	2.5	
a5	4.7	7.2	7.2		74.9		6.0			
a6					18.6	76.0		3.6		1.8
a7	3.5	6.8	9.4		8.1		72.2			
a8					2.1		3.9	64.6	13.2	16.2
a9					5.8		2.9	6.9	80.1	4.2
a10					4		0.3	2.8	3.3	89.6

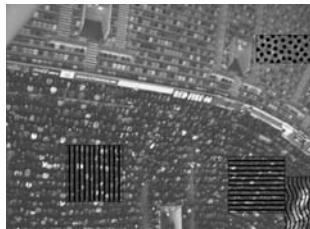
combination of 2x2 patches into feature vectors. Next, second and third processing steps are implemented as described in Sect. 2. For clustering with a Gaussian mixture model we use 15 mixture components with diagonal covariance matrices. Based on the class labels obtained by this EM-Clustering we train the model in supervised and unsupervised fashion. In both cases we compare the results with the naïve Bayesian classifier that learns directly on the hidden variables of the Gaussian mixture model.

For the supervised set up one sub-chain per class is trained using the additional information about the origin of each feature vector. In the unsupervised set up a SHMM with 16 sub-chains is trained and automatically mapping is performed using a naïve Bayesian classifier.

We train on 50% of all labelled data. The remaining 50% of the labelled data are then used to test the classification performances. As listed in Table 4b) the naïve Bayesian Classifier reaches a correct classification rate of 75.9% in total,

the supervised SHMM learning arrives at 79.8% and the unsupervised SHMM learning reaches 82.5%. It follows that, except for the FLAG class, the SHMM model reaches a better performance than the naïve Bayesian approach. The worse than Bayesian classifier rate is obtained for the FLAG-class in the supervised approach. It seems that the size of the receptive field used in this experiment was not enough to capture the full complexity of the FLAG motion pattern. To the contrary, other events such as jumping or staying, which occupy a much "smaller volume" in the spatial and temporal domain, have been correctly captured by a receptive field size of 16x16 pixels.

Furthermore, it is in the nature of the Bayesian classifier to exclusively associate emission states with class labels while the SHMM associate class labels with state transitions. Evidently, for the JUMP, GROUP, and BGROUND classes the state transitions contain more information relevant for classification. In case of the FLAG class, the Bayesian classifier associates the class label with emission states that code for a faster motion pattern and therefore induces the slightly better classification rate. The difference in performance becomes even more evident when computing the mutual information. For the Bayesian classifier this amounts to 0.67 bits and for the SHMM classifier to 1.06 bits (supervised) and 1.03 bits (unsupervised) on average. This indicates: While the Bayesian classifier mostly shows uncertain decisions, the SHMM learning seems to generate reliable class labels.



(a)

Class (Event)	Bayesian Classifier	supervised SHMM	unsupervised SHMM
JUMP	93.9	98.3	99.6
GROUP	66.0	74.1	75.6
FLAG	64.2	57.5	70.9
BGROUND	70.6	74.5	72.2

(b)

Fig. 4. (a) Annotated event maps with different motion patterns. FLAG (vertical wave pattern), JUMP (horizontal stripes), GROUP (vertical stripes) and BGROUND (dotted pattern). (b) Classification results for the supervised and unsupervised training of SHMM and for the Bayesian Classifier.

4 Conclusion

We have presented a novel approach for the unsupervised learning of motion patterns in image sequences. This approach is based on a mixture of HMMs whose hidden variables are also modelled by a Markov chain.

In our experiments we could show that SHMM model is able to learn motion sequence patterns from videos showing actions of single individuals and from surveillance videos featuring collective behaviour of large groups of people. In both cases we reached satisfactory recognition rates given the relative simplicity of SHMM model and the small amount model parameters needed for learning.

Since learning is done by computing only the forward step, our approach can deal with arbitrary long video sequences and provides the possibility to develop an on-line learning method.

Some experiments show that SHMM model might become unstable during the learning of the switching Markov chain. In order to sustainably eliminate these instabilities we can regularise our model using a Dirichlet prior for the switch transition matrix. This should yield an improved control over the diagonal values of this matrix.

Further efforts will be made to integrate SHMM model as a module in larger hierarchical systems.

References

1. Efros, A.A., Berg, A.C., Mori, G., Malik, J.: Recognizing action at a distance, vol. 2, pp. 726–733 (2003)
2. Zelnik-Manor, L., Irani, M.: Event-based analysis of video, vol. 2, pp. II–123–II–130 (2001)
3. Jhuang, H., Serre, T., Wolf, L., Poggio, T.: A biologically inspired system for action recognition. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8 (2007)
4. Casile, A., Giese, M.A.: Critical features for the recognition of biological motion. *Journal of vision* 5(4), 348–360 (2005)
5. Dean, T.: Scalable inference in hierarchical generative models. In: Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics (2006)
6. Sutskever, I., Hinton, G.: Learning multilevel distributed representations for high-dimensional sequences. *AI and Statistics* (2007)
7. Wang, P., Ji, Q.: Multi-view face tracking with factorial and switching hmm. In: Seventh IEEE Workshops on Application of Computer Vision, WACV/MOTIONS 2005, vol. 1, pp. 401–406 (2005)
8. Wu, Y., Hua, G., Yu, T.: Switching observation models for contour tracking in clutter. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings, vol. 1, pp. I–295–I–302 (2003)
9. Bayerl, P., Neumann, H.: Disambiguating visual motion through contextual feedback modulation. *Neural Comput.* 16(10), 2041–2066 (2004)
10. Bayerl, P., Neumann, H.: Disambiguating visual motion by form-motion interaction - a computational model. *International Journal of Computer Vision* 72(1), 27–45 (2007)
11. Tlapale, E., Masson, G.S., Kornprobst, P.: Motion integration modulated by form information. In: Daucè, E., Perrinet, L. (eds.) 2nd French Conf. on Computational Neuroscience, NeuroComp 2008, October 2008, pp. 245–250 (2008)
12. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39(1), 1–38 (1977)
13. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2), 257–286 (1989)
14. Gorelick, L., Blank, M., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(12), 2247–2253 (2007)
15. Ghahramani, Z., Hinton, G.E.: Variational learning for switching state-space models. *Neural Comp.* 12(4), 831–864 (2000)

Multimodal Belief Integration by HMM/SVM-Embedded Bayesian Network: Applications to Ambulating PC Operation by Body Motions and Brain Signals

Yasuo Matsuyama*, Fumiya Matsushima, Youichi Nishida,
Takashi Hatakeyama, Nimiko Ochiai, and Shogo Aida

Waseda University, Department of Computer Science and Engineering,
Tokyo, 169-8555, Japan
{yasuo,f-matsushima,y-nishida,rdsg,
n.ochai,grimoire_a1}@wiz.cs.waseda.ac.jp
<http://www.wiz.cs.waseda.ac.jp>

Abstract. Methods to integrate multimodal beliefs by Bayesian Networks (BNs) comprising Hidden Markov Models (HMMs) and Support Vector Machines (SVMs) are presented. The integrated system is applied to the operation of ambulating PCs (biped humanoids) across the network. New features in this paper are twofold. First, the HMM/SVM-embedded BN for the multimodal belief integration is newly presented. Its subsystem also has a new structure such as a committee SVM array. Another new feature is with the applications. Body and brain signals are applied to the ambulating PC operation by using the recognition of multimodal signal patterns. The body signals here are human gestures. Brain signals are either HbO₂ of NIRS or neural spike trains. As for such ambulating PC operation, the total system shows better performance than HMM and BN systems alone.

Keywords: Multimodal beliefs, Bayesian network, HMM, committee SVM array, ambulating PC.

1 Introduction

As for the realization of computational intelligence, many effective algorithms with versatile granularity have been presented [1], [2]. They enable us to handle signals with patterns for a wide variety of applications in transdisciplinary science and engineering. Various biological signals measured by sensors and cameras belong to a class of popular research targets. In order to reflect such a trend, this paper sets two targets:

* This study was supported by the “Ambient SoC Global COE Program of Waseda University” from MEXT Japan, and the “Waseda University Grant for Special Research Projects” #2009B-367.

- (1) Presentation of a new class of algorithms which provides better recognition performance.
- (2) Applications of those algorithms to new problems which comprise of Human-Machine Interfaces (HMIs). Here, “machine” stands for an ambulating PC or a biped humanoid¹.

As for item (1), the total learning and recognition system uses a Bayesian network (BN) [3]. Even though this is the case, it is known that a BN often becomes computationally intractable for practical applications. In addition, unnecessary links remain unless manual pruning is applied. Therefore, we present a BN which comprises of Hidden Markov Models (HMMs) and Support Vector Machines (SVMs) as its subsystems. This is a new structure called HMM/SVM-embedded Bayesian network. There is a new property in the subsystem, i.e., a committee Support Vector Machine array (committee SVM array). Its structure is similar to a layered neural network. This system enriches recognition classes and performance better than the BN systems, HMM systems, and SVM systems alone.

For item (2), the operation of an ambulating PC is addressed. In this problem, a human operator issues bio-signals to operate the ambulating PC across the network. The bio-signals include human motions (gestures). Others come from the brain; Near Infra-Red Spectroscopy (NIRS) or spike trains. These signals are recognized by the HMM/SVM-embedded BN as soft commands of likelihoods. This recognition step enhances machine-independence. After the hardening, solid commands are issued to the ambulating PC.

In the rest of this paper, items (1) and (2) are explained in more detail. Before delving into these sections readers are recommended to preview Fig. 3 to grasp the concept of the HMM/SVM-embedded BN, Fig. 5 for the bidirectional interaction system of the human and the ambulating PC, and Fig. 7 for experiments so that the relationships among the sections are clear.

2 HMM/SVM-Embedded Bayesian Network

2.1 Bayesian Network

The Bayesian network (BN) is an acyclic directed graph whose edges have acquired conditional probabilities [1], [3]. Fig. 1 illustrates an example which will be used in the visual recognition of objects. The top parent probability is decided by measurements of occurrence frequencies. The Conditional Probability Table (CPT) of discrete states is also estimated from conditional frequencies.

Bayesian networks can accept inputs called “evidences.” In Fig. 1, e_{index} represents evidence. If an input fixes a node probability of one, it is called hard evidence. Otherwise, inputs are called soft evidence. In the experiments in this paper, both evidence types are used. The most important property of the Bayesian network

¹ Hereafter, “ambulating PC” and “biped humanoid” will be used interchangeably. These terminologies will also be expressed simply as “humanoid” since the property of “ambulating” or “walking” is an essential ability for the machine in this paper.

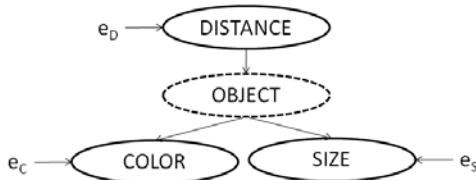


Fig. 1. A Bayesian network for object recognition

is that other node probabilities are changed by evidence injection. This computation follows the Bayes theorem. In Fig. 1, OBJECT is the state to be decided after accepting evidences. A drawback of the Bayesian network is that the number of nodes and edges are limited from the computational complexity if real-time operations are required. Instead of making a large monolithic Bayesian network, this paper devises a medium size Bayesian network which is comprised of novel and established learning gadgets as subsystems. This structure becomes equivalent to a large Bayesian network.

2.2 Committee SVM Array

A Support Vector Machine (SVM) [5] classifies input vectors. As for the SVM, there are two problems which need to be modified and improved.

- (a) Instead of a dichotomy, a multi-class decision is required in most practical applications. Unfortunately, a multi-class SVM suffers from computational overload even on a medium scale.
- (b) Given the same class of inputs, there can be many kernel types and feature extraction methods for the SVM. The top performance machine may still be unsatisfactory in performance.

To cope with (a) and (b), we devised the machine illustrated in Fig. 2. In the first layer, an array of SVMs with different possible kernels are arranged. Each SVM has a margin put into the upper layer. The upper layer renders decisions. We call

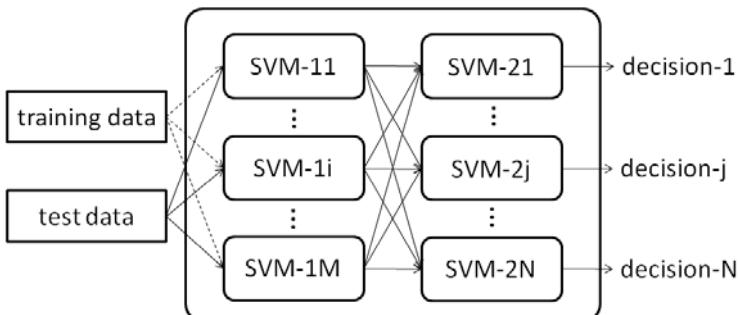


Fig. 2. Committee SVM array

this structure the committee SVM array. The committee SVM array, which is similar to a neural network, enables multi-feature, multi-kernel and multi-class decisions which are computationally heavy due to their large monolithic SVM.

2.3 Hidden Markov Models

Every Hidden Markov Model (HMM) has a representation of a BN, and vice versa [1]. But, HMM has a concise learning method. Therefore, HMM is preferred if its compact structure matches an assigned problem. As for the structure of the HMM, we adopt a simple left-to-right model. But, for multiple ones, they are placed parallel. This is because different types of signals with different patterns need to be recognized within a delay as close to real-time as possible.

2.4 Total System Configuration

In this section, the aforementioned BN, HMM and SVM systems are integrated into a total system called the HMM/SVM-embedded Bayesian network. A higher-level Bayesian network is used to glue the subsystems of BN, HMM and SVM into one machine. Such a system becomes necessary for the following reasons:

- (a) A large monolithic BN becomes too heavy to learn computationally.
- (b) A class of signals may fit to a specific learning machine. For instance, the HMM is good at the recognition of stochastic periodicities. The SVM is superior to other machines in vector classifications.
- (c) By adopting such signal-oriented subsystems, the total BN is free from learning unnecessary links.
- (d) The total BN can include nodes for unknown states to be decided after accepting evidence. This enhances the total recognition performance.

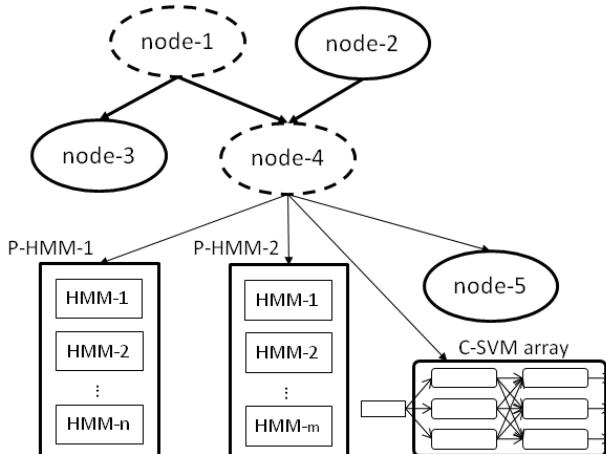


Fig. 3. HMM/SVM-embedded Bayesian network. Solid line figures receive evidence.

Fig. 3 illustrates a structure which will appear in later experiments. Note that each arrow's direction corresponds to the conditioning of the probability, and not to an output direction.

3 Human-Machine Interface

3.1 Motion Capturing

In this section, two different types of motion capturing are presented. Capturing by a Web camera is inexpensive and is regarded as a baseline method for human motion recognition. Capturing by sensors is a more advanced method which supplies richer data.

Capturing by a Web Camera: Motion capturing by a Web camera is addressed to understand how much granularity of human motions or gestures can be recognized from video images. Fig. 4 shows an illustration of a human extracted by the procedure below:

- Step 1: Compute a difference in the background.
- Step 2: Classify each pixel into binary values. Remove noise pixels by a median filter. Then, expansion and shrinking are performed. This gives an estimated cluster of pixels which is the object.
- Step 3: Region-of-Interest (ROI) is determined and redrawn.

Motion Capturing by Sensors: Motion capturing by sensors is dedicated to extract human motion. In this paper, a person performing gestures wears 11 magnetic sensors [6]. Each sensor generates a set of numerical time series of positions, angles and speeds. These raw data are then converted into the BVH format (BioVision Hierarchical format).

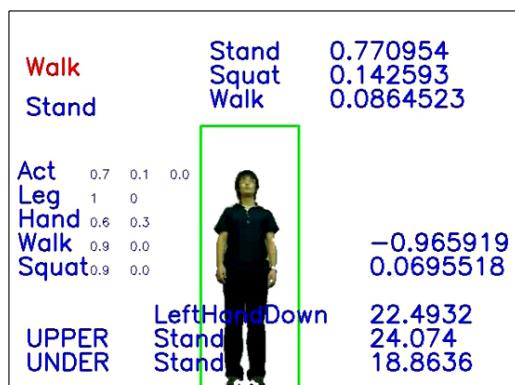


Fig. 4. Extracted human shape and belief values

In later sections, human motion will be applied to the operation of the humanoid. Since humans and humanoids are quite different in size, centroids of parts and speed, sending raw data causes malfunctioning and the collapse of humanoids. Therefore, we have to prepare a sophisticated motion recognition system comprising of the HMM/SVM-embedded Bayesian network in Fig. 3

3.2 Image Processing

The humanoid has two eyes (CCD cameras) [7] which can be utilized for shape recognition and to measure the distance from an object. This process uses color extraction and stereo matching. Then, the computed results are fed to the BN of Fig. 1 as evidence to the nodes. In later experiments, the number of object types is eight. This corresponds to the maximum ability of the cameras without auto-focusing.

3.3 Brain Signals

NIRS Measurement: For non-invading measurements of brain states, current tools are still limited because of the constraint. For real time application, possible systems are further limited. Taking this into consideration, we use an NIRS system (Near InfraRed Spectroscopy) which measures the difference of the hemoglobin level (ΔHbO_2) by probes [8]. Brain signals are generated and classified as follows:

- Step 1: A human operator tries a specific task, e.g. moving his/her eyebrows closer together.
- Step 2: Corresponding brain signal is transformed by FFT. Trends and specific noise frequencies are filtered out.
- Step 3: SVM is learned by using a set of teacher signals reflecting the assigned task.

Spike Trains: Neural spike trains provide much faster brain response than NIRS. The method used here is invading by probes. Therefore, measuring such signals *in vivo* from a human for the purpose of BMI (Brain-Machine Interface) for the humanoid is somewhat premature because of the lack of a safety measure for the subject. Despite such difficulty, help still exists. There are open databases on monkey spike trains [9]. We make use of such data for the humanoid control as follows.

- Step 1: Multiple spike trains generated by different tasks performed by the subject are provided.
- Step 2: Each spike train is used to generate an EPSP signal (Excitatory Post-Synaptic Potential).
- Step 3: EPSPs are FFT'ed. Trends and noise frequencies are filtered out.
- Step 4: HMMs are learned by using a set of teacher signals reflecting the assigned task.

It is important to note that teacher signals' tasks and humanoid tasks can be set *different*. We would rather push this conversion than to mimic the original ones. This is made possible by the existence of the recognition mechanism.

4 Operation of Ambulating PC

4.1 System Configuration

The ambulating PC, or the biped humanoid, is operated by a human operator. The HMM/SVM-embedded Bayesian network issues a series of operation commands which are the results of the bio-signal recognition. Fig. 5 illustrates the entire system which has the following configuration.

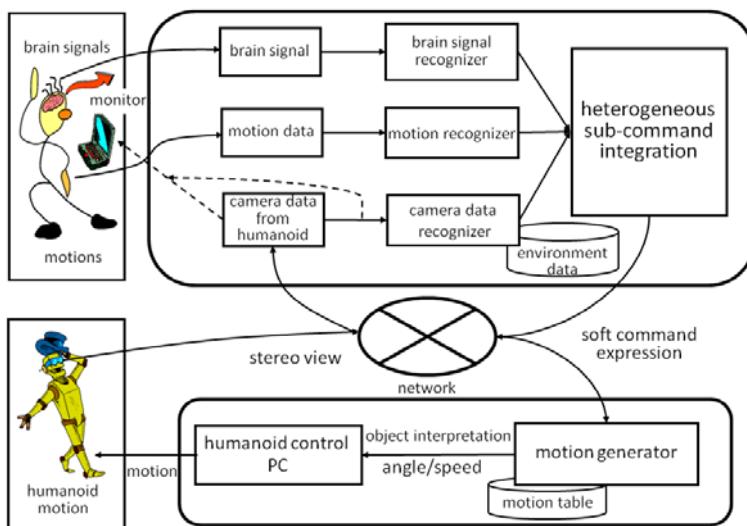


Fig. 5. Total HCI/BMI system over the network

Human Operator Side: The upper half of Fig. 5 is the site for the generation and recognition of operation signals.

- (a) Human motions are captured by systems in Section 3.1. The motion recognizer is the HMM/SVM-embedded BN in Fig. 3.
- (b) The NIRS brain signal is also measured and fed to this recognizer. Neural spike trains are preprocessed and classified as was explained in Section 3.3. This path is not drawn in Fig. 5 because *in vivo* measurement on a human brain is not yet appropriate.

- (c) The video images in the monitor which the human operator watches are sent from the humanoid's eyes across the network. These images are utilized for the machine recognition of objects and the distance measurement by the stereo matching.

- (d) Recognized results are integrated by the HMM/SVM-embedded BN.

Network: In the middle, there is a network. This is not a mere all-pass communication channel. Humans and humanoids are regarded as intelligent nodes in the network. Messages (acknowledge signals) are written and read out from a blackboard.

Ambulating PC: The ambulating PC, or the humanoid HOPA-2 [7], can be located in a different environment from the human operator, e.g. on Mars while the human operator is on Earth. Two eyes (CCD cameras) send images to the human operator so that he/she can see the same thing. It is important to note that the motion generator receives commands generated by the motion recognition. This enhances the robustness of the ambulation by absorbing the difference of human and humanoid capabilities. This method also establishes the machine independence.

4.2 Recognition Methods

Motion Recognition from Video Images: The preprocessing illustrated in Fig. 4 for the ROI extraction is applied first. As for the recognition of walking, each of the images in the 16 frames are FFT'ed. The FFT results are fed to an SVM to identify walking/non-walking. As for the recognition of non-walking patterns, the ROI was quantized to 4×8 binary meshes. Then, the mesh feature [10] is computed and fed to an HMM recognizer. Then, the following 11 motions are classified additionally; standing still, both hands-up, both hands-down, right hand-up, left hand-up, right hand-down, left hand-down, right leg-up, left leg-up, and right leg-down, and left leg-down.

Motion Recognition by Magnetic Sensors: The human operator wears 11 magnetic sensors each of which detects {three dimensional position, direction angle, rotation matrix} at the speed of 120 fps. Data are in binary streams which are converted to the BVH format before the motion recognition. The motion capture sensors increase the number of recognizable gesture patterns besides the 13 (2+11) patterns stated above. They are stretching upward, half squatting with both hands swinging, torso twisting, R/L-shoulder rotating forward and R/L-shoulder rotating backward.

One might think that such dedicated motion capturing could generate signals without noise or error so that direct transmission to the humanoid is sufficient. But, this is still not the case.

- (a) Sensors are subject to environmental noises and wobble errors.
- (b) Even if noise and wobbles were completely filtered out, a direct transmission of resulting data would cause the humanoid to fall down. This is because {weight, size, centroids of parts, speed, movable angles} are quite

different between the human and the humanoid. Therefore, the HMM/SVM-embedded BN recognizer becomes significant for the sensor system.

Brain Signal Recognition

NIRS Signals: When a human moves his/her eyebrows closer together, a specific brain pattern appears. This state can be identified by using ΔHbO_2 signal of NIRS. Noise and pulsation periodicity in such a signal are filtered out. Then, the resulting signals are applied to the SVM.

Spike Trains: On-line applications of spike trains have yet to be realized [11], [12]. In this paper, we use the spike trains from V5/MT of a monkey while it watches moving dots with different patterns [9]. The spike trains are used to generate EPSPs by leaky integrators [13], [14]. These EPSPs are smoothed by FFT, pruning, and IFFT. Then, resulting signal classes are recognized by parallel HMMs. It is important to note that the origin of the spike trains is a visual area, and not a motor area. This transducing is made possible by the recognition of patterns.

4.3 State Design

Since the humanoid is biped, falling-down is the most adverse action. The human operator and the humanoid are very different in terms of size, weight, freedom of movement, and speed. Therefore, commands by recognition might cause the humanoid to fall down if there is no matching interpretation. In other words, it is necessary to provide safety so that falling-down can be avoided. Fig. 6 illustrates such a mechanism. In this state transition diagram, intermediate states are inserted. For instance, the command of “squat” while “walk” is executed by “walk→stop→upright→squat”. Such an artifice makes it possible to avoid impossible requests to the humanoid.

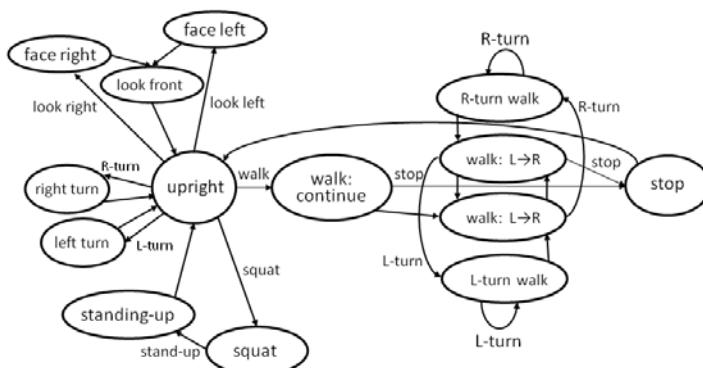


Fig. 6. Designed state transition diagram for the lower half of the body

5 Ambulating PC's Operation Using HMM/SVM-Embedded Bayesian Network

5.1 Supervised Mode Experiments: Operation by Recognized Commands

Video Recognition Method: As was explained in Section 3, a class of recognizable gestures from videos is a subset of the magnetic sensor system. This is natural since it is not possible to recognize occluded limbs. Therefore, we regard this method of video recognition system as the baseline. The variable motion segment recognition method by the HMM/SVM-embedded BN showed a recognition performance of 63% for the main gestures of the ambulating PC operations. For the upper-body-only and lower-body-only, the scores were better, 87% and 72%, respectively.

Magnetic Sensor Method: The magnetic sensor recognition system, which gives a super set of recognizable gesture patterns, showed performance of 89% for the main gestures of the ambulating PC operations. This score should be considered all right since the rest, 10%, is mainly due to the misrecognition of “right/left-leg-up and pose” from “start to walk” which is difficult to recognize even for a human. Note that a BN-only system showed performance of 76%, which proves the superiority of the HMM-embedded BN over the monolithic BN. The systems of HMM and SVM alone cannot perform this recognition.

Brain Signal Commands: The brain signal of NIRS recognized by the SVM was used to overwrite the gesture commands. Spike trains classified by the HMM are used in the same way. In the total HMM/SVM-embedded BN, nodes for NIRS and spike trains were prepared so that two different classes of brain signals could be accepted.

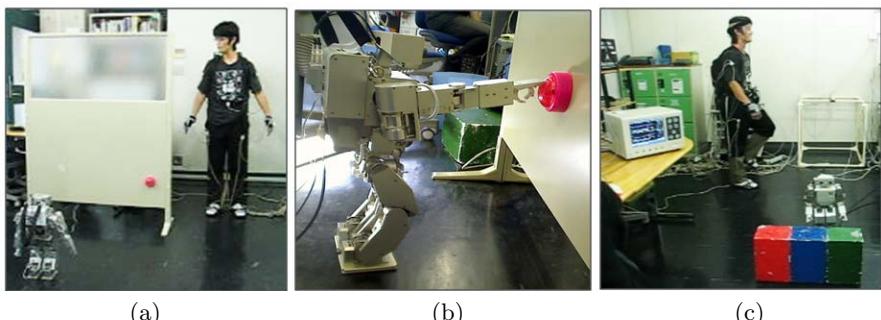


Fig. 7. Supervised mode, autonomous mode, and brain signal operation

5.2 Autonomous Mode and Total Experiment

In the total experiment, an autonomous mode was also introduced. If the humanoid finds a cue in its environment, it temporarily becomes free from the operator's commands. Instead, the humanoid tries to perform a specified task corresponding to the cue. The HMM/SVM-embedded BN provides judgement, whether the ambulating PC goes into the autonomous mode or not.

Fig. 7(a) is a scene where the human operator controls the humanoid by his gestures in the supervised mode. The operator watches images sent from the humanoid's eyes. When the humanoid finds a cue, it goes into the autonomous mode. Fig. 7(b) shows a snapshot of the humanoid trying to push a button, which is a cue, freely from the operator's commands. Fig. 7(c) is a scene where the operator sends his brain signal generated by moving his eyebrows closer together. Although the operator is walking, the brain signal overwrites all commands so that the humanoid stops. Spike trains can serve the same role in a faster response time. The important thing here is that the spike trains are from V5/MT [9]. Therefore, the spike trains that correspond to the perception of visual motions are changed to humanoid motions by the HMM/SVM-embedded Bayesian network. This is the first piece of evidence that shows mental formations can be transduced.

6 Conclusion

In this paper, the HMM/SVM-embedded Bayesian network was presented. This learning and recognition system is new and worthy of mention.

- (a) Large Bayesian networks which are computationally intractable can be avoided.
- (b) The recognition performance outperforms the BN systems, HMM sysyetms, and SVM systems alone.
- (c) The total system comprises of a novel subsystem, the committee SVM array.

In order to verify the significance of the HMM/SVM-embedded Bayesian network, we experimented on the ambulating PC control by human generated signals. The signals were gestures and NIRS brain signals as well as neural spike trains. These were successfully recognized and used as commands to the ambulating PC. Inclusion of the recognition mechanism enabled enhanced robustness and machine independence. In addition, this mechanism made it possible to transduce signals across all the senses.

References

1. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, Upper Saddle River (2003)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer Science+Business Media, LLC, New York (2006)

3. Perl, J.: Bayesian networks: A model for self-activated memory for evidential reasoning. In: Proc. 7th Conf. Cognitive Science Society, pp. 329–334 (1985)
4. Matsuyama, Y., Yoshinaga, S., Okuda, H., et al.: Towards the unification of human movement, animation and humanoid in the network. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) ICONIP 2004. LNCS, vol. 3316, pp. 1135–1141. Springer, Heidelberg (2004)
5. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
6. Ascention Technologies: Motion Star instruction manual (2000)
7. Fujitsu Automation and Michiya System: HOAP-2 instruction manual (2004)
8. Hamamatsu Photonics: NIRO 200 document, Shizuoka, Japan (2003)
9. Bair, W.: Neural signal archive, <http://www.neuralsignal.org/>
10. Umeda, M.: Recognition of multi-font printed Chinese characters. In: Proc. 6th ICPR, pp. 793–796 (1982)
11. del Pobil, A.P., Carmena, J.M., Demiris, Y.: Robotics and neuroscience. Brain-machine interfaces, and social interaction. In: Tutorial TTF6, IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan (2004)
12. Duke University and ATR: Monkey's thoughts propel robot, a step that may help humans. New York Times, January 15, 2008 (correction January 2008), <http://www.nytimes.com/2008/01/15/science/15robo.html>
13. Matsuyama, Y., Shirai, K., Akizuki, K.: On some properties of stochastic information processes in neurons and neuron populations. Kybernetik (Biological Cybernetics) 15, 127–145 (1974)
14. Matsuyama, Y.: A note on stochastic modeling of shunting inhibition. Biological Cybernetics 24, 139–145 (1976)

A Neural Network Model of Metaphor Generation with Dynamic Interaction*

Asuka Terai and Masanori Nakagawa

Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, Japan

Abstract. The purpose of this study is to construct a computational model that generates understandable metaphors of the form “A (target) like B (vehicle)” from the features of the target based on a language statistical analysis. The model outputs candidate nouns for the vehicle from inputs for the target and its features that are represented by adjectives and verbs. First, latent classes among nouns and adjectives (or verbs) are estimated from statistical language analysis. Secondly, a computational model of metaphor generation, including dynamic interaction among features, is constructed based on the statistical analysis results. Finally, a psychological experiment is conducted to examine the validity of the model.

1 Introduction

This study regards metaphor generation as a process where an expression consisting of a target (A) is modified by certain features to become a metaphorical expression of the form “target (A) like vehicle (B)” and constructs a computational model that generates understandable metaphorical expressions, represented in the form of “A (target) like B (vehicle)”. There are numerous studies that attempt to account for the process of metaphor comprehension. In such studies, a metaphor of the form “A is like B” is regarded as making good sense metaphorically if the high-salient features of the vehicle (B) match the low-salient features of the target (A)[1]. For example, there is a metaphorical expression “cheeks like apples” in Japan. In this case, “red” is a high-salient feature of “apples” and a low-salient feature of “cheeks”, so that “cheeks like apples” can be interpreted as meaning “red cheeks”. Another earlier study has pointed out that greater emotional and sensory similarity between the target and the vehicle makes metaphors more understandable[2]. In the case of “cheeks like apples”, there are other similar features between “apples” and “cheeks” in addition to “red”, such as “round” and “sweet”. And these similarities also facilitate the understandability of this metaphorical expression. If you make the expression of “cheeks like chili peppers” for “red cheeks”, because “red” is the only common feature between the target and the vehicle, the metaphor is more difficult to

* This research is supported by MEXT’s program “Promotion of Environmental Improvement for Independence of Young Researchers” and Grant-in-Aid for Scientific Research (B) (19330156).

understand. Thus, when a vehicle has other shared features of a target, rather than just the feature that is being emphasized in the metaphorical expression, the metaphor tends to be understandable. Likewise, in generating the metaphorical expression which represents “red cheeks”, it is necessary to select a vehicle that not only has the high-salient feature of “red” but also has other emotionally and sensory similar features with the target of “cheeks”, if the sense of “red cheeks” is to be understandable from the metaphorical expression.

It is worth noting that some studies have focused on the low-salient features of a target and a vehicle that are emphasized in the process of metaphor understanding; a phenomenon referred to as feature emergence [3] [4]. Feature emergence has also been described in terms of an interaction among features [5] [6] [7]. Thus, in the process of metaphor generation, we may assume some influences due to interactions among features. In the case of “red cheeks”, the expression evokes the image of “healthy” and “plump”. In contrast, “pale cheeks” evokes an unhealthy image. If a vehicle which is similar to the target in the point of “red” and “unhealthy” (e.g. “bloodstain”), is selected because not only “healthy” and “plump” but also “unhealthy” can be regarded as features of the target “cheeks” for the metaphorical expression of “red cheeks”, the metaphorical expression would not be understandable. Accordingly, it is necessary to consider influences from the interactions among features where there are emotional and sensory similarities between the target and the vehicle.

There are a few theories of metaphor generation within the field of psychology. However, some computational models of metaphor generation have also been developed [8] [9] [10]. Sako, Nakamura and Yoshida (1993) [8] constructed a computational model of metaphor generation based on a psychological experiment. However, it is not practically feasible to collect sufficient data to construct a model that sufficiently covers metaphorical expressions by their method alone, because participants cannot respond to numerous indices within a limited time. On the other hand, Kitada and Hagiwara’s (2000) model [9] is based on an electronic dictionary [11] while Abe, Sakamoto and Nakagawa’s (2006) [10] model is based on the results of statistical language analysis. These models have the advantage that they can cover many metaphorical expressions. In addition, the statistical language analysis, which does not require any manpower, is a more objective method than using an electronic dictionary, because the dictionary itself is the product of many professionals. However, both of these models fail to fully consider the interactions among features.

In order to overcome problems with previous models, the current model is constructed using a recurrent neural network based on a statistical language analysis. First, knowledge structures are estimated as the latent classes of nouns and adjectives (or verbs) using a statistical language analysis [12] for four kinds of modification patterns (frequency data for adjective-noun modifications and three kinds of verb-noun modifications). Second, the model of metaphor generation is constructed using a recurrent neural network based on the results of the statistical language analysis. Inputting a set of expressions (e.g. “transient hope” and “hope disappear”) to the model, it outputs an adequate noun for the

vehicle (e.g. “glim” (a faint light) in “hope like a glim”). Finally, a psychological experiment is conducted in order to verify the validity of the constructed model.

2 Estimation of a Knowledge Structure Using the Statistical Language Analysis

In this study, the model is constructed based on a statistical language analysis [12], which was also used in a previous study [10]. The statistical language analysis [12] with the structure similar to probabilistic latent semantic indexing (pLSI) [13] estimates latent classes among nouns and adjectives (or verbs) as a knowledge structure using four kinds of frequency data extracted for adjective-noun modifications (Adj) and three kinds of verb-noun modifications: noun(subject)-verb (S-V), verb-noun(modification) (V-M) and verb-noun(object) (V-O). These frequency data are extracted from the Japanese newspaper “MAINICHI SHIN-BUN” for the period 1993-2002 using the Japanese dependency parser CaboCha [14]. The extracted data consists of 21,671 noun types and 3,403 adjective types for adjective-noun modifications, 29,745 noun types and 22,832 verb types for verb-noun(object), 26,113 noun types and 21,487 verb types for noun(subject)-verb, and 28,451 noun types 24,231 verb types for verb-noun(modification). The extracted nouns are represented using n_i^r and the extracted adjectives or verbs are represented using a_j^r where (r refers to the kind of data set (adjective-noun modification, noun(subject)-verb (S-V), verb-noun(modification) (V-M) and verb-noun(object) (V-O) data).

The statistical method assumes that the noun n_i^r and the adjective (or verb) a_j^r co-occur through latent classes and that the co-occurrence probabilities of these terms, $P(n_i^r|a_j^r)$, can be computed using the following formula(1):

$$P(n_i^r, a_j^r) = \sum_k P(n_i^r|c_k^r)P(a_j^r|c_k^r)P(c_k^r), \quad (1)$$

where c_k^r indicates the k th latent class assumed within this method for the r type of modification data. The parameters ($P(n_i^r|c_k^r)$, $P(a_j^r|c_k^r)$, and $P(c_k^r)$) are estimated as the value that maximizes the log likelihood of the co-occurrence frequency data between n_i^r and a_j^r using the EM algorithm. The statistical language analysis is applied to each set of co-occurrence data fixing the number of latent classes at 200.

The conditional probability of the latent class c_k given the term n_i^r (noun) or a_j^r (adjective or verb) ($P(c_k|n_i^r)$ and $P(c_k|a_j^r)$) are computed using Bayes' theory, as follows:

$$P(c_k^r|n_i^r) = \frac{P(c_k^r)P(n_i^r|c_k^r)}{\sum_{k'} P(c_{k'}^r)P(n_i^r|c_{k'}^r)}, \quad (2)$$

$$P(c_k^r|a_j^r) = \frac{P(c_k^r)P(a_j^r|c_k^r)^r}{\sum_{k'} P(c_{k'}^r)P(a_j^r|c_{k'}^r)}. \quad (3)$$

The meanings of the latent classes are identified from the conditional probability of the latent class c_k^r given the adjective a_j^r and the conditional probability of

Table 1. Examples of estimated latent classes for the noun(subject)-verb (S-V) modification data which can be interpreted as a light class

	$P(c_{light}^{S-V} a_j^r)$	$P(c_{light}^{S-V} n_i^r)$	
1 blink on	0.993	red light	0.810
2 transfer	0.952	lamp	0.647
3 be burnt	0.930	glim	0.585
4 light up	0.925	illuminations	0.577
5 be lighted up	0.891	neon	0.572
6 blink	0.858	candle	0.531
7 disappear	0.761	Olympic flame	0.480
8 twinkle	0.742	electric bulb	0.455
9 flicker	0.725	torch	0.444
10 spread	0.722	chandelier	0.427

the latent class c_k^r given the noun n_h^* . For example, a latent class that can be labeled “light” is shown in Table 1

The 18,142 noun types (n_h^*) that are common to all four types modification data are represented as vectors using four kinds of conditional probabilities for latent classes given the nouns ($P(c_k^{Adj}|n_h^*)$, $P(c_k^{S-V}|n_h^*)$, $P(c_k^{V-M}|n_h^*)$, $P(c_k^{V-O}|n_h^*)$), which are computed using the four data sets, as follows:

$$V_p(n_h^*) = P(c_k^r|n_h^*), \quad (4)$$

where $V_p(n_h^*)$ indicates the p th component of the vector that corresponds to the noun n_h^* . p refers to the successive number of latent classes extracted from the four data sets. When $1 \leq p \leq 200$, r indicates the “Adj” modification and $k = p$, when $201 \leq p \leq 400$, r indicates the “S-V” modification and $k = p - 200$, when $401 \leq p \leq 600$, r indicates the “V-M” modification and $k = p - 400$, and when $601 \leq p \leq 800$, r indicates the “V-O” modification and $k = p - 600$. The dimensions of the vectors represent the latent classes. The similarity between two concepts is represented by the cosine of angles existing between concept vectors.

The meaning of the nouns (concepts) are estimated, as follows formula(5):

$$P(a_j^r|n_h^*) = \sum_k P(a_j^r|c_k^r)P(c_k^r|n_h^*). \quad (5)$$

The conditional probability of a adjective or verb (feature) given a noun (concept) ($P(a_j^r|n_h^*)$) indicates the strength of the relationships between the feature and the concept.

3 The Model of Metaphor Generation

3.1 Architecture of the Model

The model deals with the 18,142 noun types (n_h^*) that are common to all four types of modification data. The model consists of three layers: an input layer, a

hidden layer and an output layer. Sets of expressions, such as “ $a_{j_0}^{r_0} - n_{h_0}^*$ ”, “ $a_{j_1}^{r_1} - n_{h_0}^*$ ”, are input into the model. The model outputs each concept’s (noun’s) adequacy for the vehicle, which represents a set of expressions as “ $n_{h_0}^*$ (target) like B (vehicle)”.

The input layer consists of feature nodes. Each feature node indicates either an adjective or a verb. Every feature node has mutual and symmetric connections with the other feature nodes. The weight of a connection between two features for one target may differ from the weight of the connection between the same two features for another target. For example, in the case of “cheeks” (target), the connection between “red” and “healthy” will be heavily weighted. On the other hand, in the case of “eyes”, the connection between them will be less heavily weighted. The weights are estimated as follows. First, the features related to the target are selected. The meaning of the target is estimated using the conditional probability of feature given a target ($P(a_j^r | \text{target})$) and features are selected if $P(a_j^r | \text{target})$ exceeds the threshold ζ^r . These selected features and the input features ($a_{j_0}^{r_0}, a_{j_1}^{r_1}, \dots$) are related to the process of metaphor generation and the set of these features is indicated using $F(n_{h_0}^*, a_{j_0}^{r_0}, a_{j_1}^{r_1}, \dots)$. The weight for each connection between any two features, ($w_{a_j^r a_{j'}^{r'}}^{11}$), that are relating to the input target is estimated as the correlation coefficient among these features, ($\text{cor}_{a_j^r a_{j'}^{r'}}$), related to sibling concepts of the target ($n_{h_0}^*$) that are in the sibling neighborhood for a target of size s computed on the basis of similarity to the target vector. The weight for each connection between two nodes, as follows:

$$w_{a_j^r a_{j'}^{r'}}^{11} = \begin{cases} \text{cor}_{a_j^r a_{j'}^{r'}} & \text{if } a_j^r \in F(n_{h_0}^*, a_{j_0}^{r_0}, a_{j_1}^{r_1}, \dots) \text{ and } a_{j'}^{r'} \in F(n_{h_0}^*, a_{j_0}^{r_0}, a_{j_1}^{r_1}, \dots) \\ 0 & \text{else,} \end{cases} \quad (6)$$

where $w_{a_j^r a_{j'}^{r'}}^{11}$ indicates the weight for the connection between two nodes corresponding to features a_j^r and $a_{j'}^{r'}$, in the input layer.

The hidden layer consists of latent class nodes. Each node corresponds to the estimated latent class based on the statistical language analysis. There are connections from the feature nodes to the latent class nodes. The weights of these connections, ($w_{c_k^{r'} a_j^r}^{21}$), are estimated from the conditional probabilities of the latent classes given the features (adjectives or verbs), as follows:

$$w_{c_k^{r'} a_j^r}^{21} = \begin{cases} P(c_k^{r'} | a_j^r) & \text{if } r = r' \\ 0 & \text{if } r \neq r', \end{cases} \quad (7)$$

where $w_{c_k^{r'} a_j^r}^{21}$ indicates the weight for the connection from the node indicating a feature a_j^r in the input layer to the node indicating a latent class $c_k^{r'}$ in the hidden layer.

The output layer consists of concepts nodes. There are connections from the latent class nodes to the concept nodes. The weights are estimated based on the

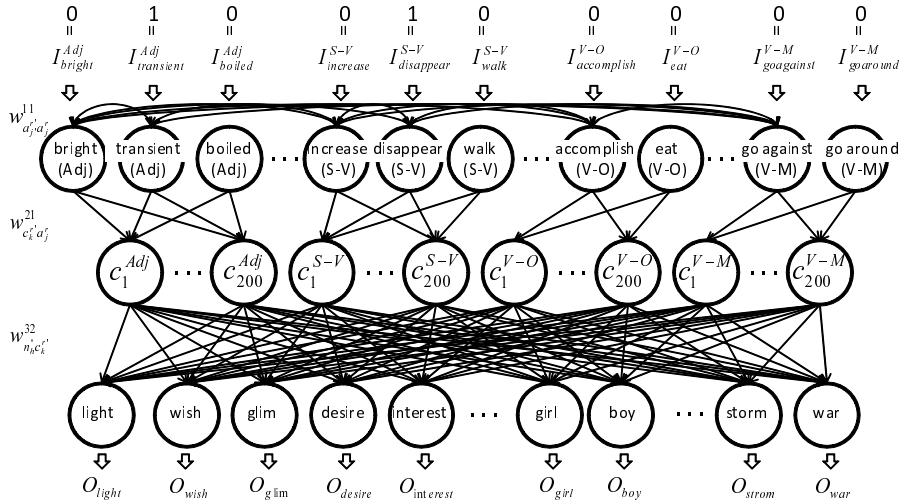


Fig. 1. Architecture of the model into which the set of “transient hope” and “hope disappear” is input. Connections whose weights are not 0 are indicated.

conditional probabilities of the latent classes given the concepts (nouns) ($w_{n'_h c'_k}^{32}$), as follows:

$$w_{n'_h c'_k}^{32} = P(c'_k | n'_h), \quad (8)$$

where $w_{n'_h c'_k}^{32}$ indicates the weight for the connection from the node meaning a latent class c'_k in the hidden layer to the node meaning a concept n'_h in the output layer.

For example, the model into which the set of “transient hope” and “hope disappear” is input is shown in Fig. 1. Only the connections whose weights are not 0 are represented. In the model, “transient (Adj) ($a_{j_0}^{r_0}$, $r_0 = Adj$)” and “disappear (S-V) ($a_{j_1}^{r_1}$, $r_1 = S-V$)” are input features. “Bright (Adj)”, “increase (S-V)”, “accomplish (V-O)” and “go against (V-M)” are features selected as being features relating to “hope” (target n'_{h_0}). Thus, there are connections among these features. On the other hand, “boiled (Adj)”, “walk (S-V)”, “eat (V-O)” and “go around (V-M) are not related to “hope”, and so these features have no interaction with other features.

3.2 Simulation of the Network

Simulation of the network consists of two steps. One is to simulate the dynamic interaction among features and the other is to simulate each the adequacy of each concept (noun) for the vehicle. For example, the set of “transient hope” and “hope disappear” is input into the model. In this case, the input features are “transient (Adj)” and “disappear (S-V)”.

First, the features nodes that correspond to “transient (Adj)” and “disappear (S-V)” have distributed input values of 1 ($I_{\text{transient(Adj)}} = 1$ and $I_{\text{disappear(S-V)}} = 1$) while all other nodes = 0. The dynamics of the network are based on the following set of simultaneous differential equations(9):

$$\frac{dx_{a_j^r}}{dt} = \exp(-\alpha t)(-x_{a_j^r}(t) + f(\beta \sum_{a_{j'}^{r'}} w_{a_j^r a_{j'}^{r'}}^{11} x_{a_{j'}^{r'}}(t) + \gamma I_{a_j^r})), \quad (9)$$

where $x_{a_j^r}(t)$ represents the activation strength of the feature node corresponding to a_j^r at time t and where the function f is a sigmoid function $\exp(-\alpha t)$ is the term for convergence which decreases according to time t . When $dx_{a_j^r}/dt = 0$, the feature node outputs $o_{a_j^r} = x_{a_j^r}(t)$. β indicates the influences of the interaction among features and γ represents the influences of input features ($a_{j_0}^{r_0}, a_{j_1}^{r_1} \dots$). If the feature a_j^r is not the input feature and the feature slightly relates to the target (namely $P(a_j^r|\text{target})$ does not exceed the threshold ζ^r), the connection weights between the node corresponding to a_j^r and the others in the input layer are 0 and the output $o_{a_j^r}$ does not activate. Even if the feature a_j^r strongly relates to the target, the output $o_{a_j^r}$ has small value on the condition that the feature slightly relates to the input features. When the feature a_j^r strongly relates to the input features and the target, the output $o_{a_j^r}$ has large value. Therefore, the output $o_{a_j^r}$ represents the input features and influences from the interactions among features where there are emotional and sensory similarities between the target and the vehicle.

Second, the output values of the concept nodes are computed using the feature node outputs, as follows:

$$O_{n_h^*} = \sum_{c_k^{r'}} w_{n_h^* c_k^{r'}}^{32} (\sum_{a_j^r} w_{c_k^{r'} a_j^r}^{21} o_{a_j^r}), \quad (10)$$

where $O_{n_h^*}$ represents the output value of the concept node corresponding to the concept n_h^* . The value represents the adequacy of the vehicle, which represents the input expression as a metaphor. The output values are computed from the input features and the interactions among features where there are emotional and sensory similarities between the target and the vehicle ($o_{a_j^r}$) through the latent classes as a knowledge structure. Thus, when the output value of a concept is high, the concept includes not only the input features (“transient (Adj)” and “disappear (S-V)”) but also emotional and sensory similarities with the target that are influenced by the interactions among features.

3.3 Model Simulation Results

In this study, the model is simulated using the parameters $s = 50$, $\alpha = \ln(10)$, $\beta = 0.3$, $\gamma = 10$, $\zeta^{\text{Adj}} = 0.00294$ (= 10/the number of adjectives), $\zeta^{\text{S-V}} = 0.00047$ (= 10/the number of verbs(S – V)), $\zeta^{\text{V-M}} = 0.00041$ (= 10/the number of verbs(V – M)), and $\zeta^{\text{V-O}} = 0.00044$ (= 10/the number of

Table 2. Simulation results of the model with interaction ($O_{n_h^*}$) and a model without interaction ($O'_{n_h^*}$) (“transient hope”, “hope disappear”). The output values are shown in parentheses.

	“transient hope”, “hope disappear”	
	the model with an interaction ($O_{n_h^*}$)	the model without an interaction ($O'_{n_h^*}$)
1	desire (0.6169)	request (0.6530)
2	fire source (0.4354)	desire (0.6525)
3	light (0.4292)	interest (0.6195)
4	glim (0.4093)	conviction (0.6029)
5	dark clouds (0.3689)	motivation (0.5991)
6	neon (0.3680)	imperative (0.5668)
7	lamp (0.3603)	inclination (0.5644)
8	delight (0.3532)	will (0.5537)
9	illuminations (0.3526)	confident (0.5503)
10	red light (0.3493)	discomfort (0.5390)

verbs($V - O$)). The simulation results for “transient hope” and “hope disappear” are shown in Table 2. For comparison, simulation results for a model of metaphor generation without interaction among the features are also shown in Table 2. The model without interaction has the algorithm similar to Abe et al.’s model [10]. The adequacies for the vehicle are computed as follows:

$$O'_{n_h^*} = \sum_{c_k^{r'}} w_{n_h^* c_k^{r'}}^{cl} \left(\sum_{a_j^r} w_{c_k^{r'} a_j^r}^{lf} I_{a_j^r} + P(c_k^{r'} | target) \right). \quad (11)$$

In the model without interaction, emotional and sensory similarities to the target are represented by inputting the image of the target $P(c_k^{r'} | target)$ into the model.

The results for both models (models with and without interaction) seem to be appropriate. However, the following psychological experiment is necessary in order to examine the validity of the model with interaction.

4 Psychological Experiment

In order to examine the validity of the model, a psychological experiment was conducted.

- Participants: 14 graduate students.
- Expressions: Two sets of expressions (“transient hope” and “hope disappear”: “heavy discussion” and “be involved in discussion”)
- Scale: 7-point scale, from 1 “Strongly disagree” to 7 “Strongly agree”.

The participants were presented with each set of expressions and the simulation results (top 10 candidates for the vehicle) for the models with and without interaction. The participants were asked to evaluate the validity of each candidate set. The results relating to the validities of the models are shown in Table 3. In

Table 3. The results of the psychological experiment. Mean rating scores are shown. Standard deviations are also indicated in parentheses.

	the input set of the model with the model without expressions	an interaction	an interaction
“transient hope”, “hope disappear”	4.29 (1.38)	2.21 (1.12)	
“heavy discussion”, “be involved in discussion”	4.50 (1.70)	4.57 (1.40)	

in the case of the model with interaction, the mean validity ratings for both sets of expressions (“transient hope” + “hope disappear” and “heavy discussion” + “be involved in discussion”) are more than 4, which equates to “neutral”. On the other hand, for the model without interaction, the mean validity rating for the set of “transient hope” and “hope disappear” is less than 3, which corresponds to “slightly disagree that the set of candidate is appropriate”.

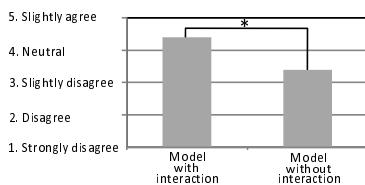


Fig. 2. The rating difference between the models with and without interaction

Furthermore, the rating difference between the two models was tested using a t-test. The results indicates that the mean validity ratings for the model with interaction (4.39) is significantly higher than the mean for the model without interaction (3.39) at 5% level ($t(27)=2.58, p<.05$) (shown in Fig. 2). This finding indicates that the model with interaction among the features is

more appropriate than the model without interaction, and so this experiment result supports the validity of the model with interaction.

5 Discussion

In this study, a computational model of metaphor generation with dynamic interaction was constructed based on data obtained through a statistical language analysis [2]. Inputting sets of expressions that consist of a feature and a target to the model, the model outputs an adequacy for the vehicle noun, which represents the input expression in the form of “A (target) like B (vehicle)”. In order to reflect the influences of interactions among features as an approach to estimating the emotional and sensory similarities between a target and a vehicle in metaphor generation, the model is constructed using a recurrent neural network with connections between the nodes that correspond to features. Compared with a model without interaction among features, which has the algorithm similar to that employed in the previous model [10], the conducted psychological experiment supports the validity of the model with interaction among features.

However, there is still some room for improvement with the psychological experiment in terms of further experiment with a wider range of expression sets. In this paper, the experiment was conducted with only two sets of expressions. In order to examine the more general validity of the model, it is necessary to

conduct further experiments using simulation results relating to many kinds of input sets. Furthermore, the results of the psychological experiment indicate that the standard deviation for each rating was more than 1-point. That implies that there are individual differences in terms of metaphor generation. It should be possible to represent such individual differences by changing the values of some of the model's parameters (e.g. the β parameter that represents the influences of the interactions among the feature and the γ parameter that represents the influences of input features). In order to clarify the relationship between individual difference in terms of metaphor generation and the parameters, we have to conduct psychological experiments in which participants evaluate multiple model simulation results obtained with various parameter settings.

References

1. Ortony, A.: Beyond Literal Similarity. *Psychological Review* 86(3), 161–180 (1979)
2. Kusumi, T.: *Hiyu no Syori Katei to Imikozo*. Kazama Syobo (1995)
3. Nueckles, M., Janetzko, D.: The role of semantic similarity in the comprehension of metaphor. In: Proc. of the 19th Annual Meeting of the Cognitive Science Society, pp. 578–583 (1997)
4. Gineste, M., Indurkhyia, B., Scart, V.: Emergence of features in metaphor comprehension. *Metaphor and Symbol* 15(3), 117–135 (2000)
5. Utsumi, A.: *Hiyu no ninchi/Keisan Moderu*. Computer Today 96(3), 34–39 (2000)
6. Terai, A., Nakagawa, M.: A Neural Network Model of Metaphor Understanding with Dynamic Interaction based on a Statistical Language Analysis; Targeting a Human-like Model. *International J. of Neural Systems* 17(4), 265–274 (2007)
7. Terai, A., Nakagawa, M.: A Corpus-Based Computational Model of Metaphor Understanding Incorporating Dynamic Interaction. In: Kürková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 443–452. Springer, Heidelberg (2008)
8. Sako, T., Nakamura, J., Yoshida, S.: Generation of Metaphorical Expressions using Surface Similarity and Emotional Evaluation between word concepts. IEICE technical report. Natural language understanding and models of communication 37(131), 17–24 (1993)
9. Kitada, J., Hagiwara, M.: Figurative Composition Support System Using Electronic Dictionaries. *Transactions of Information Processing Society of Japan* 42(5), 1232–1241 (2001)
10. Abe, K., Sakamoto, K., Nakagawa, M.: A Computational Model of Metaphor Generation Process. In: Proc. of the 28th Annual Meeting of the Cognitive Science Society, pp. 937–942 (2006)
11. EDR, Electronic Dictionary: Japan Electronic Dictionary Research Institute Ltd., Tokyo (1994)
12. Kameya, Y., Sato, T.: Computation of probabilistic relationship between concepts and their attributes using a statistical analysis of Japanese corpora. In: Proc. of Symposium on Large-scale Knowledge Resources: LKR 2005, pp. 65–68 (2005)
13. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. of the 22nd International Conference on Research and Development in Information Retrieval: SIGIR f 1999, pp. 50–57 (1999)
14. Kudoh, T., Matsumoto, Y.: Japanese Dependency Analysis using Cascaded Chunking. In: Proc. of the 6th Conference on Natural Language Learning, pp. 63–69 (2002)

Almost Random Projection Machine

Włodzisław Duch and Tomasz Maszczyk

Department of Informatics, Nicolaus Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland
{wduch,tmaszczyk}@is.umk.pl
<http://www.is.umk.pl>

Abstract. Backpropagation of errors is not only hard to justify from biological perspective but also it fails to solve problems requiring complex logic. A simpler algorithm based on generation and filtering of useful random projections has better biological justification, is faster, easier to train and may in practice solve non-separable problems of higher complexity than typical feedforward neural networks. Estimation of confidence in network decisions is done by visualization of the number of nodes that agree with the final decision.

Keywords: Neural networks, learning, random projections.

1 Introduction

The discovery of backpropagation of errors (BP) algorithm [1] for training of the multilayer perceptrons (MLPs) broke the deadlock of training non-linear systems to solve non-separable problems. However, the degree of non-separability that can be handled, measured by the k -separability index [2], is rather low. Although various version of backpropagation algorithm can deal with the XOR problem finding optimal solution for higher than the 4-bit parity problems without assuming special architecture and initialization of the network is quite hard. The original BP article was entitled “Learning internal representations by error propagation”, however these internal representations have rarely been analyzed, because they are not too informative. Neural networks that have clear neurobiological motivation create sparse, simple representation in their hidden layers [3]. Popular MLP neural networks are much simpler, they do not use internal inhibition and their only bias towards simple solutions is based on regularization [4], smoothing the mapping implemented by the network. This is not an appropriate bias for problems with complex logical structure, therefore poor generalization should be expected. Analysis of other useful biases and realistic learning targets is quite fruitful [5].

Biological neural networks solve complex learning problems inherent in optimization of behavior, creation of internal models, understanding of linguistic patterns. Creating algorithms capable of solving problems of similar complexity is an important challenge and is needed to open the doors for a new generation of ambitious machine learning applications. Backpropagation of errors is hard to justify from the neurobiological perspective. Algorithms that are biologically plausible and should be able to learn complex functions are therefore of great interest. Deep belief networks are one

interesting candidate [6]. Maass et al. [7] have stressed that high-dimensional dynamics allows for real-time computing without stable states. Instead of attractor neural networks that require extensive training his Liquid State Machines can use much simpler high-dimensional dynamics that corresponds to very complex microcircuit resonances. Perceptrons may work then as readout neurons, extracting in real-time stable information from transient internal states formed by such high dimensional system. Kernel machines [8] work on similar principle, implicitly projecting data into high-dimensional spaces, where decision borders become flat and separation by linear hyperplanes is relatively easy.

The almost Random Projection Machine (aRPM) algorithm presented here is based on the following inspirations. Learning to read, learning multiplication table or similar tasks takes weeks, although brain plasticity of children is higher than adults. Synaptic learning is usually rather slow and it takes a long time before new connections will develop. Yet even old people may quickly learn and remember many things after a single exposure. The amount of synaptic learning must thus be rather limited. Neurons in association cortex form strongly connected microcircuits found in cortical minicolumns, resonating with different frequencies when an incoming signal $X(t)$ appears. A perceptron neuron “observing” the activity of thousands of microcircuits in these minicolumns learns to react to specific signals around particular frequency. However, resonators do not get excited when overall activity (weighted combination of inputs $\mathbf{W} \cdot \mathbf{X}$) is high, but rather react when specific levels of activity are reached, selecting only signals from some soft interval $G(\mathbf{W} \cdot \mathbf{X})$. If these signals are correlated with important activity its contribution is taken into account, otherwise the signal is not used.

The feature space created in this way is based on those combinations of inputs that have been found interesting for some task, and thus have some meaning and interpretation. These features are not learned but selected from random projections, with new features added if they show interesting correlations with some aspect of the problem being solved. In classification this would mean a subset of vectors from a single class, some of which have not yet been captured by too many other features and thus carry interesting information. In fact this model is not too far from the original Selfridge Pan-demonium architecture [9], where demons, representing interesting observations, shout to influence decisions of demons that are higher in the hierarchy.

In the next section aRPM algorithm is formally introduced, including relations to the research on random projections. Section three presents empirical tests and comparisons with standard machine learning methods, and the last section some conclusions.

2 Almost Random Projections

Presentation of new input activates large number of microcircuits in the cortex, but competition and local inhibition will finally leave only a small number of the most active circuits that provide relevant information. They provide several views on the same data, in each case discovering a particular angle and projecting a group of similar (from this particular angle) cases, while cutting off the remaining cases from the projection. A simple threshold neuron may then read out the level of activation of specific circuits, estimating familiarity of the presented item by activation proportional to the number

of clusters from each category that this item excites. Similar idea has been used in the liquid state machines [2] designed to analyze spatio-temporal patterns. Many random oscillators are postulated, projecting the signal into highly dimensional space, and a threshold neuron is used to read out the activity of the column and discriminate between different categories. In this paper random filters are used and those that find something interesting are selected to contribute to the output.

Multi-layer perceptron (MLP), the most successful neural network model, is based on a perceptron model, or a neuron that performs soft threshold logic operation using weighted sum of input signals [10]. This is a rough but useful abstraction of activity of a single biological neuron. Logical threshold neurons, for various noisy input signal distributions concentrated around some average values, estimate conditional probabilities that change in a sigmoidal way, depending on the strength of the signal [11]. Perceptrons may thus be seen as logical devices operating on noisy data. Many random perceptrons form a hidden layer that projects the data into high-dimensional space.

Two important ideas come from such neurocognitive inspirations. First, many views of the same item should be considered, generating interesting transformations $T_i(X)$ that involve non-local projections $\mathbf{W}_i \cdot \mathbf{X}$. Such projections are filtered through localized functions $T_i(X) = G_i(\mathbf{W}_i \cdot \mathbf{X})$ discovering useful features specific to a given category. The number of features should not be fixed, as they are dynamically generated until there is sufficient information to make decision. The interplay between local and global analysis has been missing in neural networks and other types of machine learning algorithms. Transformations $T_i(X)$ map input cases to one-dimensional clusters that should be either relatively pure or at least partially discriminative, excluding some categories [12]. A single large projected cluster is sufficient for categorization if there is no strong competition, but some redundancy should be preferred. The winner-takes-most mechanism of biological networks should be approximated to make final decision based on memberships in projected clusters [3]. It is surprising that so far neural networks took only the simplest inspirations from biology.

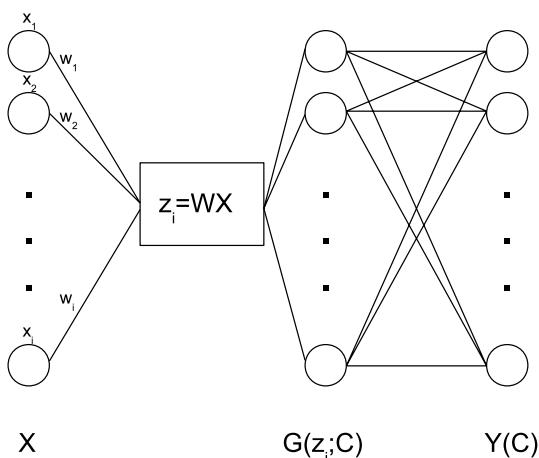


Fig. 1. Network structure of the aRPM algorithm

Algorithm 1. aRPM

Require: Flag all vectors \mathbf{X} as “new”.

```

1: for  $i = 0$  to  $N_{rep}$  do
2:   Randomly assign weights  $\underline{w}_i, w_i \in [-1, 1]$ .
3:   Generate new projection  $z_i = \underline{w}_i \mathbf{X}$ .
4:   Analyze  $p(z_i|C)$  distributions to determine “interesting” clusters.
5:   Add them as new features  $G(z_i; C)$ , or class-labeled hidden network nodes.
6:   Sum the activity of hidden node subsets for each class to calculate network outputs
 $y(C|\mathbf{X}) = \sum_i G(z_i; C)$ .
7:   Remove flag “new” from all vectors that reach  $y(C|\mathbf{X}) \geq \beta$ .
8: end for
   Validate the network.
9: if Accuracy does not increase then
10:  return network.
11: else
12:  goto 1
13: end if
```

The aRPM algorithm inspired by the ideas mentioned above has only a few parameters (see Algorithm 1). First, a relevance index [13] is applied to determine if the projected cluster is interesting, taking into account only “new” vectors, that is those that have not been already covered more than β times by other clusters (if $\beta = 1$ new nodes should cover training vectors only once). Boosting-like variants may be considered [14], but here only the simplest version is tested. Any filter based on information indices, purity or other criterion is suitable. Second, to justify adding new features (attaching hidden node to the output) new clusters that give rise to features should not be too small, covering at least α fraction of all vectors, and at least one new vector. Intervals $G(z_i; C)$ that extract clusters from projections may for example be modeled by a difference of two logistic functions, providing a soft trapezoidal function [15], but below only a simple [min,max] intervals have been used. The number of repetitions N_{rep} has been set here to 10.

The weights in this algorithm are randomly selected, and their values do not change – no additional learning of weights is needed, in contrast to all other methods for neural network training. In some feedforward network models learning is restricted only to the linear output layer [16], but here it is replaced by a simple addition of appropriate inputs. The aRPM simply generates sufficient number of random weights and selects new useful features $G(z_i; C)$. Each node corresponds then to a perceptron capturing clusters that may be surrounded by samples from other classes. To make final decision aRPM uses winner-takes-most mechanism.

Many variants of basic aRPM algorithm are possible and will be presented in a longer paper. New features may be used in a Naive Bayes type of estimation, but then one should avoid redundant nodes, while in the additive model the more nodes are excited by a given vector the better. There is no reason why all clusters should be pure, although projections that overlap with existing ones may be discouraged to generate more interesting views on the data. Additional parameter γ may determine the threshold for relevance of the new feature, for example the purity level of each new cluster. Another

possibility is to introduce weights proportional to the size of the cluster (number of vectors that pass through a particular filter), or use linear discrimination on the activity of the hidden nodes (this is more costly). $G(z_i; C)$ may reflect class-dependent probability density.

3 Illustrative Examples

The usefulness of aRPM algorithm has been evaluated on two artificial and four real datasets downloaded from the UCI Machine Learning Repository [17] and from [18] (Leukemia). A summary of these datasets is presented in Table 1. Artificial 8 and 10-bit parity datasets have been selected because they are very difficult to analyze correctly by standard MLPs, Support Vector Machines or other machine learning algorithms. The four other datasets are standard examples of benchmark type and are used here to enable typical comparison of different learning methods. Leukemia has 7129 dimensions and it would be quite easy to get perfect results with such a large space, therefore 100 best features from a simple Fischer Discriminant Analysis (FDA) ranking index have been used [13]. Vectors with missing values have been removed (6 vectors from Heart disease dataset, and 16 from Wisconsin cancer), although it is quite easy to use projections based only on features that have been defined (formally undefined features should turn off nodes that use it, but it is enough to give it sufficiently low value).

Table 1. Summary of datasets used for comparison of aRPM algorithm with other methods

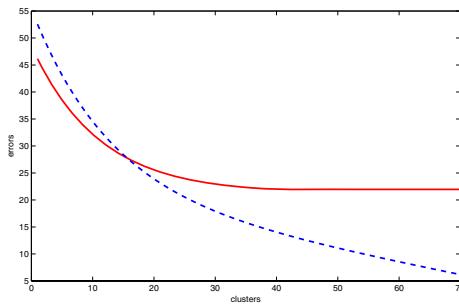
Title	#Features	#Samples	#Samples per class		Source
Parity8	8	256	128 even	128 odd	artificial
Parity10	10	1024	512 even	512 odd	artificial
Leukemia	100	72	47 ALL	25 AML	[18]
Heart	13	297	160 absence	137 presence	[17]
Wisconsin	10	683	444 benign	239 malignant	[19]
Liver	6	345	145 C_1	200 C_2	[17]

To compare aRPM with 4 popular classification methods 10-fold crossvalidation tests have been repeated 10 times and average results collected in Table 2 with accuracies and standard deviations for each dataset. Additionally in each column the complexity of the generated models have also been noted: for C4.5 size of the tree, for kNN the number of nearest neighbours, for SVM the number of support vectors, and for MLP and aRPM the number of hidden nodes. Only linear SVM has been used as for these dataset results obtained with Gaussian kernel are not better. Parameters of all classifiers have been optimized. In case of aRPM pure clusters were enforced, with the minimum number of vectors in each cluster set as 1% of all vectors for the Heart and Wisconsin datasets, for the Leukemia at least 10 vectors have been required, and for the parity very large clusters were enforced, for the 10-bit parity over 200 elements.

High-dimensional parity problem is very difficult for most classification methods. Many papers have been published on special neural models for parity functions, and

Table 2. Accuracy results

Dataset	Method				
	C4.5	kNN	MLP	SVM	aRPM
Parity8	31.6 ± 1.3 (1)	100 ± 0 (17)	94.1 ± 2.1 (17)	32.4 ± 4.4 (230)	99.2 ± 1.6 (12)
Parity10	40.4 ± 1.6 (1)	100 ± 0 (21)	89.2 ± 12.3 (21)	39.1 ± 6.5 (920)	99.5 ± 0.9 (12)
Leukemia	82.6 ± 8.3 (5)	97.2 ± 1.6 (2)	95.8 ± 3.6 (52)	98.7 ± 3.9 (15)	96.1 ± 8.6 (19)
Heart	77.8 ± 2.1 (33)	81.8 ± 6.6 (45)	79.5 ± 1.3 (8)	81.5 ± 1.3 (94)	78.3 ± 4.2 (43)
Wisconsin	94.7 ± 2.0 (21)	97.0 ± 1.7 (5)	94.2 ± 0.2 (6)	96.3 ± 2.1 (49)	97.9 ± 1.6 (30)
Liver	65.8 ± 2.2 (51)	62.0 ± 1.1 (44)	67.5 ± 3.1 (5)	69.2 ± 10.3 (236)	61.1 ± 5.1 (47)

**Fig. 2.** Typical convergence curve, showing errors as a function of the number of nodes (Heart dataset); dashed line - training errors, solid line - test errors

the reason is quite obvious. Linear separation cannot be easily achieved because this is a k -separable problem that should be separated into $n+1$ intervals for n bits [220]. This is a very interesting example showing that aRPM solves quite easily difficult problems in almost perfect way even when most standard classifiers fails. Although kNN may also work perfectly well it requires $k > 2n$ for n -bit parity to overcome the influence of the nearest neighbors.

Reduced Leukemia dataset is classified by aRPM at 96%, significantly better than C4.5 result and at the similar level as all other systems. For Cleveland Heart data the new algorithm gives about $78 \pm 4\%$ accuracy, with the base rate of 54%. This is not significantly different from other classifiers because variance is rather high, although SVM has some advantage here, with over 100 support vectors creating rather complicated model. Wisconsin breast cancer dataset is classified by aRPM with higher accuracy then other classifiers, and relatively small variance $98 \pm 2\%$. For Liver dataset variance of most methods is quite high, but SVM may have some advantage, other classifiers do not give significantly different results.

These results are by no means the limits of aRPM algorithm as α and β parameters have not been optimized and $\gamma = 0$ was taken. How many hidden nodes should be created? If α is small more nodes are created, giving higher training accuracy, but clusters covered by these nodes have to be smaller, or less pure, so some learning of optimal values is needed. The simplest version used here is very fast as it does not perform any

learning, except for setting the interval in one-dimensional projections. Typical convergence with respect to the number of nodes (Fig. 2) is quite fast and monotonic, quickly saturating. For example for the Heart data accuracy saturates at 78% for about ≈ 43 nodes.

aRPM classifier allows for easy estimation of confidence in the results. This is seen in Fig. 3 scatterplots. The network has two linear outputs and their value, for binary activations of hidden nodes, is simply an integer number between 0 and the number of nodes for each class. Each plot presents output of aRPM model trained inside cross-validation and then applied to all dataset, thus showing test and training errors. Most vectors activate only nodes from the correct class, some of them as many as 8. Large pure clusters show high-confidence predictions. Some vectors are rejected and fall into (0,0) cluster; they may be assigned to a default majority class. In case of Heart one test vector excites as many as 5 nodes from the wrong class, showing that it may be an outlier. For each cluster confidence factor is equal to the purity of this node estimated in crossvalidation.

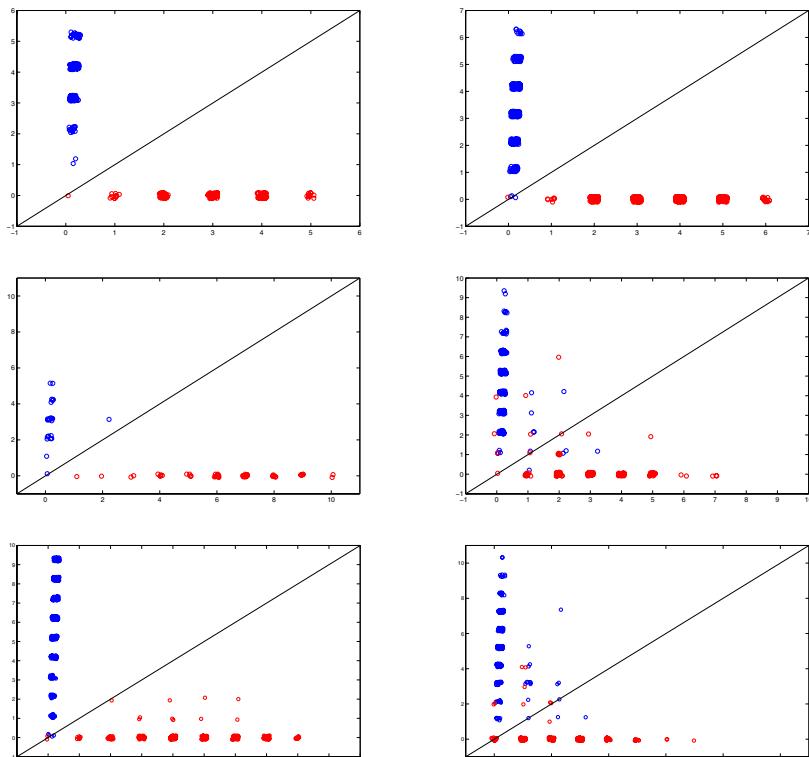


Fig. 3. Output of aRPM algorithm for training and test data, top row: Parity8 and Parity10, middle row: Leukemia and Heart, bottom row: Wisconsin and Liver

4 Discussion and New Directions

Neurocognitive informatics draws inspirations from neurobiological processes responsible for learning. So far only a few general inspirations have been used in computational intelligence: threshold neurons that perform parallel distributed processing, organized in networks. Even with our limited understanding of the brain many more inspirations may be drawn and used in practical learning and object recognition algorithms. Arguments for biological plausibility of random projections rather than slow learning mechanisms have been presented and simplest version of almost Random Projection Machine tested. Surprisingly, results of a model that does not perform any optimization are on benchmark problems at least as good as MLPs, and on parity problem which is quite difficult to learn for MLPs and SVMs are almost perfect.

Many variants of aRPM will be discussed in a longer paper: they improve results by optimizing minimal number of vector per cluster, adding impure clusters and enforcing minimal number of new vectors that have not been correctly classified, changing hard-limit intervals into more soft-window filters, and setting the threshold for similarity of new projections before new nodes are created. The brain does not use fixed number of features, as most pattern recognition algorithms do, but starting from a small number of features actively searches for new, most discriminative features that neural filters may provide. Objects are recognized using different features that characterize them. Thus feature selection and construction is not separable from the actual process of categorization and learning. This is easily incorporated into our algorithms by using subsets of all available original features to create new hidden nodes. If a very large (or small) number is inserted for the unknown value nodes that use this feature will be inactive while nodes that do not use it will provide normal activations. Visualization of outputs and the ability to estimate confidence in predictions made by such classifiers is a very useful feature of all variants of these algorithms.

The final goal of learning is to categorize, but the intermediate representations are also important. Finding interesting views on the data, or constructing interesting information filters, is the most important thing. Each filter does its own feature selection or feature weighting. Instead of using networks with fixed number of inputs systems that actively sample data, trying to “see it” through their filters, are needed. Once they have sufficient information to categorize data structures they have done their job. This opens the way to new algorithms that may learn from objects that have diverse structures, including many missing values. It is much easier to achieve non-linear separability in the hidden layers of neural networks than linear separability [5]. If the structure of non-linear mapping that creates image of data is known it may be then analyzed and understood. The most important part for good generalization in learning systems is to create large clusters, as small clusters are not reliable and will be washed out by neural noise. The learning process is greatly simplified by changing the goal of learning to easier target and handling the remaining nonlinearities with well defined structure.

Random projections facilitate rapid learning, but in biology rapid learning is followed by slow learning that perfects the function. Learning to increase usefulness of individual nodes to increase purity/separation of hidden nodes may follow initial creation of a functional network. Projection pursuit with Quality of Projected Clusters index [21] may be used for discovery of interesting views, and it should be very interesting

to use it as an algorithm for deep belief networks [6] that are trained using Restricted Boltzmann Machines. All these approaches create interesting hidden representation of the data.

References

1. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L. (eds.) *Parallel Distributed Processing: Explorations in Microstructure of Cognition. Foundations*, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
2. Duch, W.: k -separability. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) *ICANN 2006. LNCS*, vol. 4131, pp. 188–197. Springer, Heidelberg (2006)
3. O'Reilly, R., Munakata, Y.: *Computational Explorations in Cognitive Neuroscience*. MIT Press, Cambridge (2000)
4. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford (1995)
5. Duch, W.: Towards comprehensive foundations of computational intelligence. In: Duch, W., Mandziuk, J. (eds.) *Challenges for Computational Intelligence*, vol. 63, pp. 261–316. Springer, Heidelberg (2007)
6. Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. *Neural Computation* 18, 381–414 (2006)
7. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14, 2531–2560 (2002)
8. Schölkopf, B., Smola, A.: *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge (2001)
9. Selfridge, O.G.: Pandemonium: A paradigm for learning. In: Blake, D.V., Uttley, A.M. (eds.) *Proceedings of the Symposium on Mechanisation of Thought Processes*, pp. 511–529. HM Stationery Office, London (1959)
10. Haykin, S.: *Neural Networks - A Comprehensive Foundation*. Maxwell MacMillian Int., New York (1994)
11. Duch, W.: Uncertainty of data, fuzzy membership functions, and multi-layer perceptrons. *IEEE Transactions on Neural Networks* 16, 10–23 (2005)
12. Duch, W., Adamczak, R., Hayashi, Y.: Eliminators and classifiers. In: Lee, S.Y. (ed.) *7th International Conference on Neural Information Processing (ICONIP)*, Dae-jong, Korea, pp. 1029–1034 (2000)
13. Duch, W.: Filter methods. In: Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. (eds.) *Feature extraction, foundations and applications*, pp. 89–118. Physica Verlag/Springer, Heidelberg (2006)
14. Schapire, R., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. *Machine Learning* 37, 297–336 (1999)
15. Duch, W., Jankowski, N.: Survey of neural transfer functions. *Neural Computing Surveys* 2, 163–213 (1999)
16. Huang, G.-B., Chen, L., Siew, C.K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks* 17, 879–892 (2006)
17. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
18. Golub, T.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286, 531–537 (1999)

19. Wolberg, W.H., Mangasarian, O.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In: Proceedings of the National Academy of Sciences, U.S.A., vol. 87, pp. 9193–9196 (1990)
20. Grochowski, M., Duch, W.: Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 180–189. Springer, Heidelberg (2007)
21. Grochowski, M., Duch, W.: Projection Pursuit Constructive Neural Networks Based on Quality of Projected Clusters. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 754–762. Springer, Heidelberg (2008)

Optimized Learning Vector Quantization Classifier with an Adaptive Euclidean Distance

Renata M.C.R. de Souza and Telmo de M. Silva Filho

Centro de Informatica - CIn / UFPE, Av. Prof. Luiz Freire, s/n,

Cidade Universitaria CEP: 50740-540 - Recife - PE, Brasil

{rmcrs, tmsf}@cin.ufpe.br

Abstract. This paper presents a classifier based on Optimized Learning Vector Quantization (optimized version of the basic LVQ1) and an adaptive Euclidean distance. The classifier furnishes discriminative class regions of the input data set that are represented by prototypes. In order to compare prototypes and patterns, the classifier uses an adaptive Euclidean distance that changes at each iteration but is the same for all the class regions. Experiments with real and synthetic data sets demonstrate the usefulness of this classifier.

1 Introduction

Learning Vector Quantization (LVQ) is a prototype-based algorithm family proposed by Kohonen [6] that has been widely studied and applied in various areas, due to its simplicity and efficiency. The basic LVQ aims to find discriminant regions in the input data space represented by prototypes or codebook vectors. The learning algorithm starts by assigning a subset of prototypes for each pattern class in an ad-hoc way, and interactively it searches for prototypes that are neighbors to patterns. The prototypes are updated according to the classification result performed such that the nearest neighbor rule minimizes the average expected misclassification probability. When the iterations settle down, the updated prototypes should be close to the training patterns in their classes. The class of a new pattern is the same as the class of the closest prototype.

The LVQ algorithm family consists of several modifications of the basic LVQ procedure aiming to achieve best approximation of Bayes optimal decision boundaries, faster or more robust convergence, or incorporation of more flexible metrics ([1], [4] - [6], [8]). The distance between data points, determined by the chosen metric, plays a key role in LVQ and clustering algorithms. In addition, the partitional clustering [7] as well as LVQ provide a partition of the input data into classes or regions which aim to optimize a criterion (for example, the sum of intra-class inertias in k-means algorithm and average expected misclassification probability in LVQ). However, if these methods assume a static model of distance, this can make them ineffective to adequately capture characteristics of the classes or class regions in situations when the data set contains classes of different shapes and sizes or equal shape but different volumes.

In [2] were introduced partitional clustering algorithms in which the distance used to compare clusters and their prototypes changes at each iteration but is the same for all prototypes and in [3] are presented partitional fuzzy clustering methods based on this

type of distance, called adaptive distance. The importance of this distance is in the fact that it takes into account the intra-structure of the clusters and in this way the clustering algorithm is able to recognize classes of different shapes and sizes.

The main contribution of this paper is to introduce a LVQ classifier with adaptive distance. Here, the learning rule is based on the Optimized Learning Vector Quantization (OLVQ1 - an optimized version of the LVQ1 introduced by Kohonen [6]) and the search of the nearest prototype is performed assuming an adaptive Euclidean distance. The goal is to use a distance that takes into account characteristics of the regions represented by prototypes and to achieve better performance in comparison with the original OLVQ1 that uses the (non-adaptive) standard Euclidean distance.

The paper is organized as follows: Section 2 presents the optimized learning vector quantization classifier based on an adaptive Euclidean distance. Section 3 describes a performance analysis considering a synthetic data set with overlapping classes. The performance is based on the prediction accuracy that is assessed in the framework of a Monte Carlo simulation with 100 replications of this data set. Section 4 shows a performance analysis considering three real data sets. In this analysis, the prediction accuracy is assessed by the ten-fold cross validation method. Finally, Section 5 gives the concluding and final remarks.

2 OLVQ-AD Classifier

This section presents the optimized learning vector quantization classifier with adaptive distance (here called OLVQ-AD). The basic idea is to integrate an adaptive distance with an incremental learning vector quantization. Thus, the learning process has two optimization stages. In first stage, prototypes are modified according to the matching between patterns and their nearest prototypes. The matching is determined based on an adaptive Euclidean distance that takes into account the dispersion of the regions with respect to their prototypes. In the second one, the adaptive Euclidean distance is updated according to a criterion function measuring the fitting between the regions and their prototypes.

2.1 The Optimized Learning Vector Quantization Algorithm (OLVQ1)

Suppose there are K classes labelled $1, \dots, K$. Let $\Omega = \{(\mathbf{x}_i, y_i)\}$ ($i = 1, \dots, n$) be a training data set. Each pattern i is described by a vector of p quantitative values $\mathbf{x}_i = (x_i^1, \dots, x_i^p)$ and a discrete quantitative valued y_i that takes values in discrete set $\{1, \dots, K\}$.

The OLVQ is an iterative algorithm that aims to look for a prototype subset (a set of points in the quantitative feature space) for each class, such that training points attract prototypes of the correct class and repel other prototypes. An initial set of prototypes is chosen from the training set. Prototypes are moved to try to achieve better classification of the training data set by the 1-nearest neighbour rule based on a distance between prototypes and training patterns.

Let $\{(\mathbf{w}_m, y_m)\}$ ($m = 1, \dots, M$) be a set of prototypes, each one represented by a vector p quantitative values $\mathbf{w}_m = (w_m^1, \dots, w_m^p)$ and a discrete quantitative value

y_m that takes values in discrete set $\{1, \dots, K\}$. Assume that a subset of prototypes is assigned to each class k ($k = 1, \dots, K$) such that the total number of prototypes is M .

Given a pattern \mathbf{x}_i to be processed and let

$$c = \arg \min \{d(\mathbf{x}_i, \mathbf{w}_m) = \sum_{j=1}^p (x_i^j - w_m^j)^2\}$$

be the index of the prototype that lies closest to \mathbf{x}_i according to the distance d .

The process of learning in the OLVQ1 consists of updating the position of \mathbf{w}_c according to the following equations:

$$\begin{aligned}\mathbf{w}_c(t+1) &= \mathbf{w}_c(t) + \alpha_c(t)[\mathbf{x}_i - \mathbf{w}_c(t)] \text{ if } \text{class}(\mathbf{x}_i) = \text{class}(\mathbf{w}_c) \\ \mathbf{w}_c(t+1) &= \mathbf{w}_c(t) - \alpha_c(t)[\mathbf{x}_i - \mathbf{w}_c(t)] \text{ if } \text{class}(\mathbf{x}_i) \neq \text{class}(\mathbf{w}_c)\end{aligned}\quad (1)$$

where $0 < \alpha_c(t) < 1$ is the corresponding learning rate to the prototype w_c . It can be constant or decrease monotonically with time and is determined as:

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{1 + s(t)\alpha_c(t-1)} \quad (2)$$

where $s(t) = +1$ if \mathbf{x}_i and \mathbf{w}_c belongs to the same class, and $s(t)=-1$ if \mathbf{x}_i and w_c belong to different classes.

2.2 Adaptive Distance

The OLVQ-AD classifier proposed in this paper involves the construction of a fixed number of prototypes for each class based on the nearest-neighbor rule and an adaptive distance that takes into account the dispersion of the class regions. In this way, the algorithm is able to find discriminant class regions of different shapes and sizes. By processing the training pattern data set, a criterion function based on the nearest-neighbor rule can be defined as:

$$J = \sum_{i=1}^n d_A(\mathbf{x}_i(t), \mathbf{w}_c(t))\delta_{ci} \quad (3)$$

where d_A is an adaptive distance between a pattern $\mathbf{x}_i(t)$ and a prototype $\mathbf{w}_c(t)$ at iteration t such that c is the index of the nearest prototype to \mathbf{x}_i ($m = 1, \dots, M$) and $\delta_{ci} = 1$ if $\mathbf{x}_i(t)$ has the same class of $\mathbf{w}_c(t)$ otherwise $\delta_{ci} = 0$.

The criterion above represents the sum of the distances between patterns and prototypes that have the same class label. When the Euclidean distance is used, an adaptive version of this distance is as follows:

$$d_A(\mathbf{x}_i(t), \mathbf{w}_c(t)) = \sum_{j=1}^p \lambda^j (x_i^j(t) - w_c^j(t))^2 \quad (4)$$

The adaptivity of d_A is expressed by the vector of weights $\boldsymbol{\lambda} = (\lambda^1, \dots, \lambda^p)$ that represents the information of the region dispersion.

From the Lagrange Multiplier method, the weight vector $\lambda = (\lambda^1, \dots, \lambda^p)$, which minimizes the criterion in equation (3) under $\lambda^j > 0$ and $\prod_{j=1}^p \lambda^j = 1$, is updated according to the following expression

$$\lambda^j = \frac{\{\prod_{h=1}^p (\sum_{i=1}^n (x_i^h(t) - w_c^h(t))^2 \delta_{ci})\}^{\frac{1}{p}}}{\sum_{i=1}^n (x_i^j(t) - w_c^j(t))^2 \delta_{ci}} \quad (5)$$

2.3 The Algorithm

The OLVQ-AD classifier has two optimization steps. The first step aims to find a subset of prototypes (codebook vectors) into each class, minimizing the average expected misclassification probability. The second step aims to find an weighted Euclidean distance minimizing a criterion function that measures the dissimilarity between prototypes and patterns.

The schema of the OLVQ-AD algorithm is given as follow:

1. **Initialization**
 - 1.1 Choose distinct M patterns of the training data set Ω randomly, producing the initial subset of prototypes $\{\mathbf{w}_1(t), \dots, \mathbf{w}_M(t)\}$ at step $t=0$.
 - 1.2 For $m = 1$ to M do $\alpha_m(t) = 0.1$.
 - 1.3 For $j = 1$ to p do $\lambda^j = 1$.
2. **Prototype update step:** For $i = 1$ to n choose a pattern of Ω randomly without reposition
 - 2.1 Define the winning prototype w_c such that

$$c = \arg \min_{m=1, \dots, M} d_A(\mathbf{x}_i, \mathbf{w}_m)$$

$$2.2 \text{ If } \text{class}(\mathbf{x}_i) = \text{class}(\mathbf{w}_c) \text{ do } \mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \alpha_c(t)[\mathbf{x}_i - \mathbf{w}_c(t)]$$

$$2.3 \text{ If } \text{class}(\mathbf{x}_i) \neq \text{class}(\mathbf{w}_c) \text{ do } \mathbf{w}_c(t+1) = \mathbf{w}_c(t) - \alpha(t)[\mathbf{x}_i - \mathbf{w}_c(t)]$$

2.4 Update

$$\alpha_c(t+1) = \frac{\alpha_c(t)}{1 + s(t)\alpha_c(t)}$$

where $s(t) = +1$ if \mathbf{x}_i and \mathbf{w}_c belong to the same class, and $s(t)=-1$ if \mathbf{x}_i and \mathbf{w}_c belong to different classes

3. **Distance update step:** For $j = 1, \dots, p$, compute λ^j with equation (5).
4. **Stopping criterion:** If for all $\alpha_m(t) \leq 0.00005$ ($m = 1, \dots, M$) then STOP, otherwise go to 2.

A drawback in developing incremental prototype set building method is how to generate an initial prototype set. In order to solve this problem, a Monte Carlo simulation with a fixed number of different initializations for the prototype set is applied. For each repetition, a set of updated prototypes is obtained after the convergence of the algorithm and the average of the updated prototypes among these repetitions is calculated. The goal is to obtain best values for the update prototypes and to improve the performance of the classifier.

2.4 Classification Rule for New Patterns

Let ω be a new pattern, which is candidate to be assigned to one of the K classes, and its corresponding description given by the continuous feature vector $\mathbf{x}_\omega = (x_\omega^1, \dots, x_\omega^p)$. Remember that from the learning step the prototypes $\mathbf{w}_1, \dots, \mathbf{w}_M$ are computed.

The classification rule is defined as follow:

1. Find m such that

$$d_A(\mathbf{x}_\omega, \mathbf{w}_m) \leq d_A(\mathbf{x}_\omega, \mathbf{w}_g), \forall g \in \{1, \dots, M\} \quad (6)$$

2. Assign ω to class(\mathbf{w}_m)

3 Experiments on Synthetic Data Sets

Experiments with an artificial quantitative data set in \mathbb{R}^2 showing class overlapping and a corresponding performance analysis of the proposed classifier are considered in this section. The idea is to achieve an evaluation of the OLVQ-AD algorithm that uses an adaptive Euclidean distance in comparison with the OLVQ algorithm that uses the standard Euclidean distance. The evaluation is carried out based on the classification accuracy that was measured through the error rate of classification.

In this experiment, 100 replications of the data set with identical statistical properties are obtained and for each one training (75% of the original data set) and test (25% of the original data set) sets are randomly generated. The estimated error rate of classification corresponds to the average of the error rates found among the 100 replicates of test set.

The data set has 800 points scattered among four clusters of equal sizes (200 points) and ellipsis shapes of different volumes. Each class in this quantitative data set was drawn according to two independent normal distributions according to the following parameters:

- a) Class 1: $\mu_1 = 45, \mu_2 = 30, \sigma_1^2 = 100, \sigma_2^2 = 9$;
- b) Class 2: $\mu_1 = 70, \mu_2 = 38, \sigma_1^2 = 81, \sigma_2^2 = 16$;
- c) Class 3: $\mu_1 = 45, \mu_2 = 42, \sigma_1^2 = 100, \sigma_2^2 = 16$;
- d) Class 4: $\mu_1 = 42, \mu_2 = 20, \sigma_1^2 = 81, \sigma_2^2 = 9$;

Figure 11 displays the data set showing class overlapping.

As mentioned initially, 100 replications of this data set are performed in the framework of a Monte Carlo simulation in order to estimate the error rate of classification. Within this framework, 10 repetitions to choose the initial prototype set were also carried out in the framework of another Monte Carlo simulation. The values of the average (in %) \pm standard deviation of the error rate of classification for the OLVQ-AD and OLVQ algorithms were, respectively: 11.18 ± 2.142 ; and 13.71 ± 2.583 . As expected for this data set, the average error rate of the OLVQ classifier is greater than that of the OLVQ-AD classifier.

The comparison between the OLVQ and OLVQ-AD classifiers is achieved by the Student's t-test for independent samples at a significance level of 5%. Let μ_1 and μ_2 be

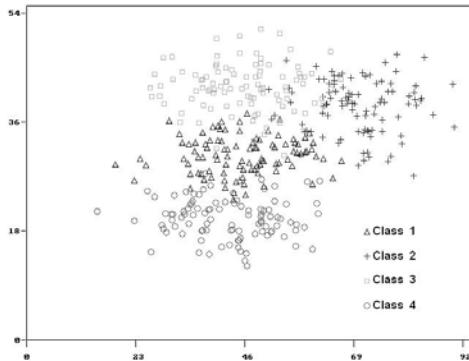


Fig. 1. Data set with four classes

the average of the error rate for the OLVQ and OLVQ-AD classifiers, respectively, the (null and alternative) hypothesis are:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 > \mu_2$$

By applying this test, the observed value of the test statistic following a Student's t distribution with 198 degrees of freedom was 7.55. This value shows that the OLVQ-AD classifier outperformed the OLVQ one.

4 Application with Real Data Sets

The OLVQ and OLVQ-AD classifiers were applied to the iris, wine and pima data sets. These data sets can be found in the UCI machine learning repository <http://www.ics.uci.edu/mlearn/MLRepository.html>. The comparison criterion used was the classification error rate obtained by the stratified cross-validation partitioning method on 10-folds. 10 repetitions of this procedure were carried out in order to estimate the classification error rate in the framework of a Monte Carlo simulation. Thus, the estimated error rate of classification corresponded to the average of the error rates found among the 100 replicates test set. For each iteration of this Monte Carlo procedure, 10 repetitions to choose the initial prototype set were also carried out in the framework of another Monte Carlo simulation.

4.1 Iris Data Set

This data set consists of three types (classes) of iris plants: Iris setosa, Iris versicolor and Iris virginica. These classes have 50 instances each. One class is linearly separable from the other 2; the latter are not linearly separable from each other. Each class is described by 4 real valued attributes. These attributes are: sepal length, sepal width, petal length and petal width.

The OLVQ and OLVQ-AD classifiers were applied to this data set adopting five prototypes for each class. The values of the average (in %) \pm standard deviation of the error rate of classification for the OLVQ-AD and OLVQ classifiers were, respectively: 3.93 ± 4.454 ; and 4.87 ± 5.513 . From these values, we have that the average error rate of the OLVQ algorithm is greater than that of the OLVQ-AD algorithm.

Student's t-test for independent samples at a significance level of 5% were applied and the observed value of the statistic tests following a Student's t distribution with 198 degrees of freedom was 0.13. This result pointed out that the null hypothesis can not be rejected and we can conclude that for this data set the OLVQ classifier was as good as the OLVQ-AD classifier.

4.2 Wine Data Set

This data set consists of three types (classes) of wines produced in the same region in Italy but derived from three different cultivars. The classes (1, 2 and 3) have, respectively, 59, 71 and 48 instances. Each wine is described by 13 real valued attributes representing the quantities of 13 constituents found in each of the three types of wines. These attributes are: 1) Alcohol, 2) Malic acid, 3) Ash, 4) Alkalinity of ash, 5) Magnesium, 6) Total phenols, 7) Flavonoids, 8) Non-flavonoid phenols, 9) Proanthocyanins, 10) Colour intensity, 11) Hue, 12) OD280/OD315 of diluted wines and 13) Proline.

The OLVQ-AD and OLVQ classifiers were applied to this data set considering three prototypes for each class. The values of the average (in %) \pm standard deviation of the error rate of classification for the OLVQ-AD and OLVQ classifiers were, respectively: 4.87 ± 4.679 ; and 28.22 ± 9.652 . These values showed that for this data set the OLVQ-AD classifier was considerably superior to the OLVQ classifier.

4.3 Pima Data Set

The National Institute of Diabetes and Digestive and Kidney Diseases of the NIH originally owned the PIMA Indian Diabetes Database (PIDD). The database has 768 patients each with 8 valued attributes. There are 500 non-diabetic patients for class 0 and 268 diabetic ones for class = 1. The quantitative variables are: number of times pregnant, 2-hour OGTT plasma glucose, diastolic blood pressure, triceps skin fold thickness, 2-hour serum insulin, BMI, diabetes pedigree function and age.

The OLVQ-AD and OLVQ classifiers were applied to this data set considering five prototypes for each class. The values of the average (in %) \pm standard deviation of the error rate of classification for the OLVQ-AD and OLVQ classifiers were, respectively: 25.87 ± 5.025 ; and 30.58 ± 5.584 . As in the Iris and Wine data sets, the average error rate of the OLVQ was greater than that of the OLVQ-AD.

Student's t-test for independent samples at a significance level of 5% was also applied and the observed value of the statistic test following a Student's t distribution with 198 degrees of freedom was 6.27. This result shows that for this data set the OLVQ-AD classifier performed better than the OLVQ classifier.

5 Conclusions

In this paper, a classifier based on Optimized Learning Vector Quantization and an adaptive distance was presented. The classifier furnishes class regions represented by prototypes and uses an adaptive Euclidean distance that changes at each iteration of the learning algorithm. The advantage of this distances is that the algorithm is able to recognize class regions of different shapes and sizes and to have better performance in comparison with the original OLVQ that uses the standard Euclidean distance.

In order to compare the proposed OLVQ-AD classifier with the standard OLVQ classifier, experiments with synthetic and real data sets were carried out. The accuracy of the results furnished by these classifier methods was assessed by the error rate of classification. Concerning the results of the error rate, the OLVQ-AD classifier with an adaptive Euclidean distance proposed in this paper was superior to OLVQ classifier that uses the standard Euclidean distance.

Acknowledgments. The authors would like to thank CNPq (Brazilian Agency) for its financial support.

References

1. Biehl, M., Ghosh, A., Hammer, B.: Learning vector quantization: The dynamics of winner-takes-all algorithms. *Neurocomputing* 69, 660–670 (2006)
2. Diday, E., Govaert, G.: Classification Automatique avec Distances Adaptatives. R.A.I.R.O. Informatique Computer Science 11(4), 329–349 (1977)
3. De Carvalho, F.A.T., Tenório, C.P., Cavalcanti Jr., N.L.: Partitional Fuzzy Clustering Methods Based on Adaptive Quadratic Distances. *Fuzzy Sets and Systems* 157(21), 2833–2857 (2006)
4. Duda, R.O., Hart, P.E., Stock, D.G.: Pattern Classification. John Wiley, Chichester (2000)
5. Hammer, B., Villmann, T.: Generalized relevance learning vector quantization. *Neural Networks* 15, 1059–1068 (2002)
6. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Berlin (2001)
7. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1998)
8. Sánchez, J.S., Marqués, A.I.: An LVQ-based adaptive algorithm for learning from very small codebooks. *Neurocomputing* 69, 922–927 (2006)

Efficient Parametric Adjustment of Fuzzy Inference System Using Error Backpropagation Method

Ivan da Silva and Rogerio Flauzino*

University of São Paulo, Department of Electrical Engineering,
CP 359, CEP 13566.590, São Carlos, SP, Brazil

{insilva,raflauzino}@sc.usp.br

<http://laips.sel.eesc.usp.br>

Abstract. This paper presents a new methodology for the adjustment of fuzzy inference systems, which uses technique based on error backpropagation method. The free parameters of the fuzzy inference system, such as its intrinsic parameters of the membership function and the weights of the inference rules, are automatically adjusted. This methodology is interesting, not only for the results presented and obtained through computer simulations, but also for its generality concerning to the kind of fuzzy inference system used. Therefore, this methodology is expandable either to the Mandani architecture or also to that suggested by Takagi-Sugeno. The validation of the presented methodology is accomplished through estimation of time series and by a mathematical modeling problem. More specifically, the Mackey-Glass chaotic time series is used for the validation of the proposed methodology.

Keywords: Fuzzy systems, tuning algorithm, error backpropagation.

1 Introduction

The design of fuzzy inference systems comes along with several decisions taken by the designers since is necessary to determine, in a coherent way, the number of membership functions for the inputs and outputs, and also the specification of the fuzzy rules set of the system, besides defining the strategies of rules aggregation and defuzzification of output sets. The need to develop systematic procedures to assist the designers has been wide because the trial and error technique is the unique often available in many cases [1]. At present time, there are several researchers engaged in studies related to the design techniques involving fuzzy inference systems [2][3][4][5].

This paper presents an approach for tuning fuzzy inference systems based on unconstrained optimization techniques using the error backpropagation method. The objective is to minimize an energy function associated with the fuzzy inference system, which can be explicated as the mean squared error between the

* The authors gratefully acknowledge the financial support, under grants 06/58850-6 and 07/03727-8, received from FAPESP.

output of the fuzzy system and the desired results, similar to the artificial neural networks with supervised training. Therefore, it is observed that the proper specification of the energy function is fundamental to apply the adjustment method correctly.

In general terms, for applications involving system identification and fuzzy modeling, it is convenient to use energy functions that express the error between the desired results and those provided by the fuzzy system. An example is the use of the mean squared error or normalized mean squared error as energy functions.

In the absence of a tuning set, such as happens in parameters adjustment of a process controller, the energy function can be defined by functions that consider the desired requirements of a particular design [6], i.e., maximum overshoot signal, setting time, rise time, undamped natural frequency, etc.

Using this new approach from the definition point of view, the fuzzy system can be designed as a three layers model. Each one of these layers represents the tasks performed by the fuzzy inference system such as fuzzification, fuzzy rules inference and defuzzification. The adjustment procedure proposed in this paper is performed through the adaptation of its free parameters, from each one of these layers, in order to minimize the energy function previously specified.

In principle, the adjustment can be made layer by layer separately. Thus, the routine of fuzzy inference system tuning acquires a larger flexibility when compared to the training process used in artificial neural networks. Simulation results are presented to validate the approach proposed.

2 Aspects of Fuzzy Inference Systems

In the last years it has been observed a wide and growing interest in applications involving fuzzy logic. These applications include since consumer products, such as cameras, video camcorders, washing machines and microwave ovens, even industrial applications as control of processes, medical instrumentation and decision support systems [10].

The fuzzy inference systems can be treated as methods that use the concepts and operations defined by the fuzzy set theory and by fuzzy reasoning methods [11]. Basically, these operational functions include fuzzification of inputs, application of inference rules, aggregation of rules and defuzzification, which represents the crisp outputs of the fuzzy system [12]. At present time, there are several researchers engaged in studies related to the design techniques involving fuzzy inference systems.

The first type of design technique of fuzzy inference system has its focus addressed to enable the modeling of process from their expert knowledge bases, where both antecedent and consequent terms of the rules are always fuzzy sets, offering then a high semantic level and a good interpretability capacity [13]. However, the applicability of this technique in the mapping of complex systems composed by several input and output variables has been an arduous task, which can produce as inaccurate results as poor performance [11,14].

The second type of design technique of fuzzy inference system can be identified as being those that incorporate learning, in an automatic way, from data that are representing the behavior of the input and output variables of the process. Therefore, this design strategy uses a collection of input and output values obtained from the process to be modeled, which differs of the first design strategy, where the fuzzy system was defined using only the expert knowledge acquired from observation on the respective system. In a generic way, the methods derived from this second strategy can be interpreted as being composed by automatic generation techniques of fuzzy rules, which use the available data for their adjustment procedures (or training).

Among the main approaches belonging to this second design strategy, it has been highlighted the ANFIS (Adaptive-Network-based Fuzzy Inference Systems) algorithm proposed by [12], which is applicable to the fuzzy architectures constituted by real polynomial functions as consequent terms of the fuzzy rules, such as those presented by [15] and [16]. The more recent approaches, such as those proposed by [17][18][19], are also belonging to this design strategy.

However, the representation of a process through these automatic architectures can implicate in interpretability reduction in relation to the created base of rules, whose consequent terms are expressed in most of the cases by polynomial functions, instead of linguistic variables [20].

Thus, the development of adjustment algorithms of fuzzy inference systems, which the consequent terms of the fuzzy rules are also represented by fuzzy sets, has been widely motivated.

3 Formulation of the Proposed Adjustment Method

Considering the operational functions performed by the fuzzy inference systems, it is convenient to represent them by a three-layers model. Thus, a fuzzy inference system can be represented by the sequential composition of three layers, i.e., input layer, inference layer and output layer.

The input layer has functionalities of connecting the input variables (coming from outside) with the fuzzy inference system, performing their respective fuzzifications through proper membership functions. In the inference layer of the fuzzy rules, the input fuzzified variables are combined among them, according to defined rules, using as support the operations defined by the fuzzy theory. The resulting set of this aggregation process is then defuzzified to produce the fuzzy inference system output. The aggregation and defuzzification process of the fuzzy system output are both made by the output layer. It is important to observe, concerning to the output layer, that although it performs the two processes above described, it is also responsible for storing the membership functions of the output variables. As illustration, Fig. 1 shows the proposed multilayer model, which is constituted by two inputs and one output, having three fuzzy rules in its inference layer.

In the following subsections further details will be presented about how fuzzy inference systems can be represented by a three-layers model.

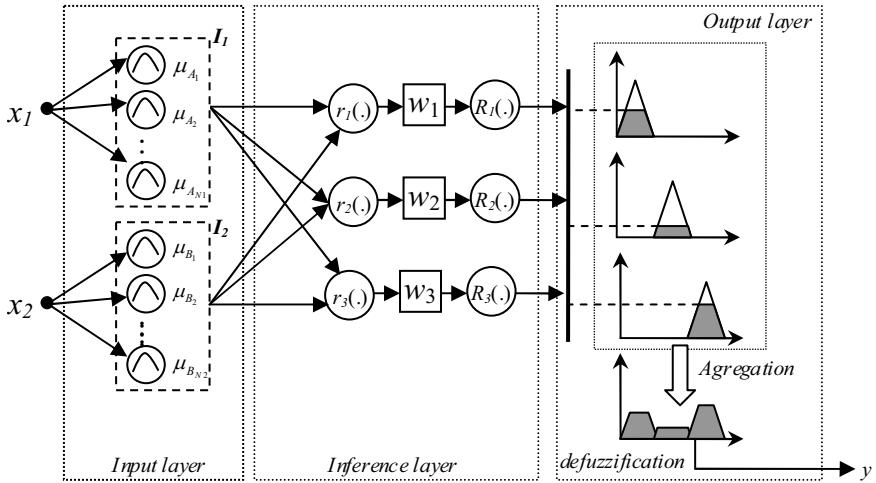


Fig. 1. Fuzzy inference system composed by two inputs and one output

3.1 Input Layer

The inputs fuzzification has the purpose of determining the membership degree of each input related to the fuzzy sets associated with each input variable. To each input variable of the fuzzy system can be associated as many fuzzy sets as necessary. In this way, let a fuzzy system constituted by only one input with N fuzzy sets, the output of the input layer will be a column vector with N elements, which are representing the membership degrees of this input in relation to those fuzzy sets. If we define the input of this fuzzy system with a unique input x , then the output of the input layer will be the vector I_1 represented by:

$$I_1(x) = [\mu_{A_1}(x) \ \mu_{A_2}(x) \ \dots \ \mu_{A_N}(x)]^T . \quad (1)$$

where $\mu_{A_k}(\cdot)$ is the membership function defined to input x , which is referring to the k -th fuzzy set associated with it. The generalization of the input layer concept for a fuzzy system having p input variables can be achieved if we consider each input being modeled as a sub-layer of the input layer. Taking into account this consideration, the output vector of the input layer $I(x)$ is then defined by:

$$I(x) = [I_1(x_1)^T \ I_2(x_2)^T \ \dots \ I_p(x_p)^T]^T . \quad (2)$$

where x_i is the i -th input of the fuzzy system and $I_k(\cdot)$ is the k -th vector of membership functions associated with the input x_k . In Fig. 1 is illustrated the input layer for a fuzzy system composed by two inputs, which are mapped by the vectors I_1 and I_2 .

There are several membership functions that can be used in the proposed approach. One of the necessary requisites for those functions is that they are normalized in the closed domain $[0,1]$.

3.2 Inference Layer

The inference layer of a fuzzy system has the functionality of processing the fuzzy inference rules defined for it. Another functionality is to provide a knowledge base for the process.

In this paper, the fuzzy inference system has initially all the possible inferred rules. Therefore, the tuning algorithm has the task of weighting the inference rules. The weighting of the inference rules is a proper way to represent the most important rules, or even to allow that conflicting rules are related to each other without any linguistic completeness loss. Thus, it is possible to express the i -th fuzzy rule as follows:

$$R_i(I(x)) = w_i r_i(I(x)) . \quad (3)$$

where $R_i(.)$ is the function representing the fuzzy weighting of the i -th fuzzy rule, w_i is the weight of the i -th fuzzy rule and $r_i(.)$ represents the fuzzy value of the i -th fuzzy rule. In Fig. 1, it is shown the composition involving $r_i(.)$ and $R_i(.)$ for the three fuzzy rules belonging to the inference layer.

3.3 Output Layer

The output layer of the fuzzy inference system aims to aggregate the inference rules as well as the defuzzification of the fuzzy set generated from the aggregation of these inference rules.

Besides the operational aspects, the aggregation and defuzzification methods must consider the requisites of hardware performance in order to reduce the computational effort needed for processing the fuzzy system. In this paper, the output layer of the inference system is also adjusted. The adjustment of this layer occurs in a similar way to that used for the input layer of the fuzzy system. As example, an illustration representing the procedures involved with the output layer is also shown in Fig. 1.

4 Adjustment of the Fuzzy Inference System

Let a fuzzy system with two inputs, each one composed of three Gaussian membership functions, with a total of five inference rules, and having an output defined by two Gaussian membership functions. It is known that, for each Gaussian membership function, two free parameters should be considered, i.e., the mean and the standard deviation. Consequently, the number of free parameters of the input layer is 12. For each inference rule, a weighting factor has been associated, resulting a total of 5 free parameters in the inference layer. In relation to the output layer, the same considerations used for the input layer are valid. Therefore, four free parameters are associated with the output layer.

Thus, the mapping f between the input space x and the output space y may be defined by:

$$y = f(x, mf_{In}, w, mf_{Out}) . \quad (4)$$

where mf_{In} is the parameter vector associated with the input membership functions, w is the weight vector of the inference rules, and mf_{Out} is the parameter vector associated with the output membership functions. Therefore, mf_{In} , w and mf_{Out} represent the free parameters of the fuzzy system, which can be rewritten as follows:

$$y = f(x, \Theta) . \quad (5)$$

where Θ is the vector resulting from concatenation of the free parameters involved with the fuzzy system, i.e.

$$\Theta = [mf_{In}^T, w^T, mf_{Out}^T] . \quad (6)$$

The energy function to be minimized, considering the fixed tuning set $\{x, y\}$, is defined by:

$$\xi \equiv \xi_{(x,y)}(\Theta) . \quad (7)$$

where ξ represents the energy function associated with the fuzzy inference system f .

4.1 Unconstrained Optimization Techniques

Let an energy function $\xi_{(x,y)}(\Theta)$ differentiable in relation to free parameters of the fuzzy inference system. Thus, the objective is to find an optimum solution Θ^* subject to:

$$\xi(\Theta^*) \leq \xi(\Theta) . \quad (8)$$

Therefore, we can observe that to satisfy the condition expressed in (8), it is necessary to solve an unconstrained optimization problem to obtain the solution Θ^* , which is given by:

$$\Theta^* \equiv \arg \min_{\Theta} \xi(\Theta) . \quad (9)$$

The condition that expresses the optimum solution in (9) can also be rewritten as follows:

$$\nabla \xi(\Theta^*) = 0 . \quad (10)$$

where ∇ is the gradient operator defined by:

$$\nabla \xi(\Theta) = \left[\frac{\partial \xi}{\partial \Theta_1}, \frac{\partial \xi}{\partial \Theta_2}, \dots, \frac{\partial \xi}{\partial \Theta_m} \right]^T . \quad (11)$$

There are several techniques used to solve unconstrained optimization problems. A detailed description of these methods can be found in [7]. The selection of the most proper method is related to the complexity associated with the energy function. For example, the Gauss-Newton method for unconstrained optimization can be more applicable in problems where the energy function is defined by:

$$\xi(\Theta) = \frac{1}{2} \sum_{i=1}^m e^2(i) . \quad (12)$$

where $e(i)$ is the absolute error in relation to the i -th tuning pattern.

In this paper, a derivation of the Gauss-Newton method is used for tuning fuzzy inference system, which is defined by the expression following:

$$\Theta_{next} = \Theta_{now} - \frac{1}{2} (J^T J)^{-1} g . \quad (13)$$

where g is the gradient of ξ expressed in (11) and J is the Jacobean matrix of e defined in (12). The optimization algorithm used was the Levenberg-Marquardt method [8], which can efficiently handle ill-conditioned matrices $J^T J$ by altering equation (13) as follows:

$$\Theta_{next} = \Theta_{now} - \frac{1}{2} (J^T J + \lambda I)^{-1} g . \quad (14)$$

The calculation of the matrices J and the vectors g were performed through the finite differences method.

5 Methodology and Results

Using the adjustment methodology presented in this paper, a fuzzy inference system of Mandani type was developed with objective to predict the Mackey-Glass time series [9], which is defined by:

$$\frac{dx(t)}{dt} = -b \cdot x(t) + \frac{a \cdot x(t - \tau)}{1 + x(t - \tau)^c} . \quad (15)$$

where the values of the constants are usually assumed as $a = 0.2$, $b = 0.1$ and $c = 10$. The value for the delay constant τ was 17. The tuning set was constituted by 500 patterns. The input variables of the fuzzy inference system were four, which correspond to values $x(t - 18)$, $x(t - 12)$, $x(t - 6)$ and $x(t)$. As output variable was adopted $x(t + 6)$.

The fuzzy inference system was defined having 4 fuzzy sets attributed to each input variable and also to the output variable. A total of 64 inference rules have been used in the inference process.

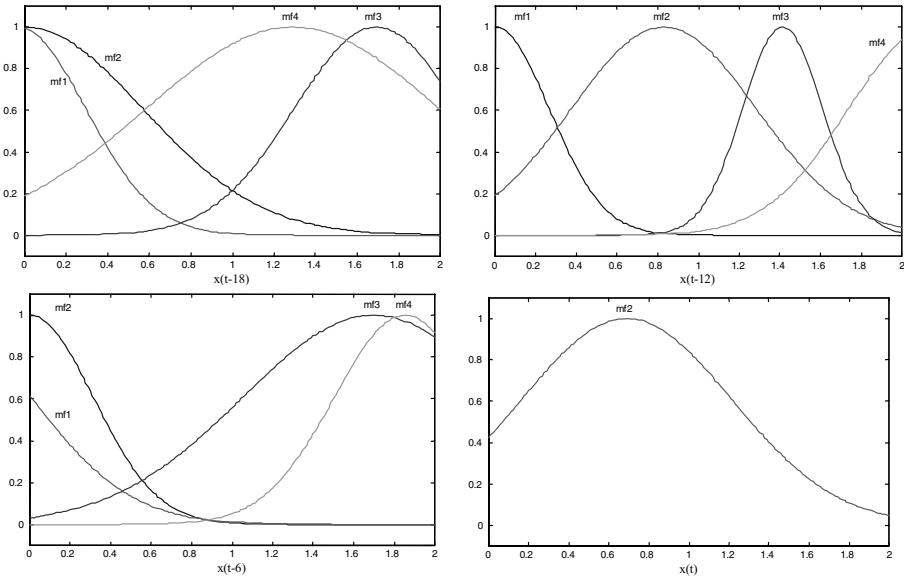
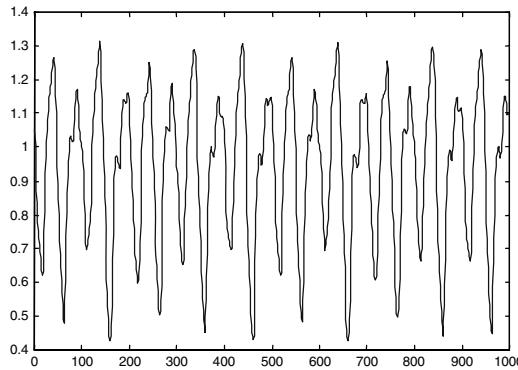
The energy function of the system was defined as being the mean squared error between the desired values $x(t + 6)$ and the estimated values $\bar{x}(t + 6)$, i.e.

$$\xi(\Theta) = \frac{1}{L} \sum_{i=1}^L [x_i(t + 6) - \bar{x}_i(t + 6)]^2 . \quad (16)$$

where L is the number of data used in the tuning process ($L = 500$).

After minimization of (16), the membership functions of the fuzzy inference system were adjusted as illustrated in Fig. 2.

In Fig. 3 is presented the prediction results provided by the fuzzy inference system for 1000 sample points. The mean squared error of estimation for the proposed problem was 0.000598 with standard deviation of 0.0245.

**Fig. 2.** Input membership functions**Fig. 3.** Estimation of the fuzzy inference system for the Mackey-Glass series

For comparison, it was developed a fuzzy inference system adjusted by the ANFIS (Adaptive Neural-Fuzzy Inference System) [12]. This fuzzy inference system was composed by 10 membership functions for each input, being the knowledge database constituted by 10 rules. The mean squared error of estimation was 0.000165 with standard deviation of 0.0041.

The proposed methodology was also used in other example in order to model a two-dimensional *sinc* function defined by:

$$z = \text{sinc}(x, y) = \frac{\sin(x) \cdot \sin(y)}{x \cdot y} . \quad (17)$$

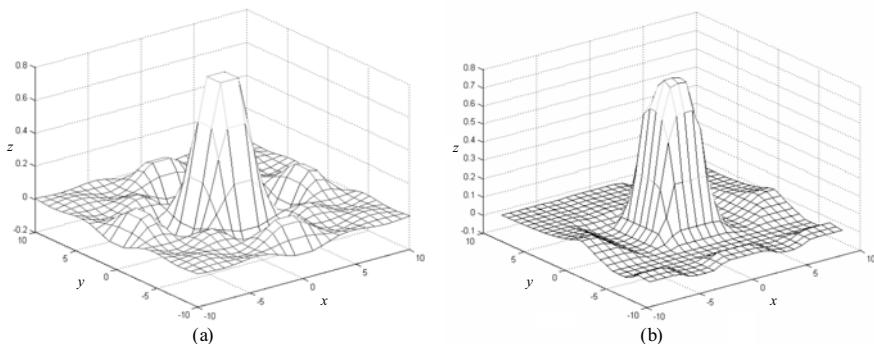


Fig. 4. Tuning data (a) and reconstructed surface (b)

From uniformly distributed grid points into the input range $[-10,10] \times [-10,10]$ of (17), 225 tuning data pairs were obtained. The fuzzy inference system used here contains 11 rules, with 8 membership functions assigned to input variable x , 7 membership functions assigned to input y and 3 membership functions assigned to output z . The tuning data and reconstructed surface are illustrated in Fig. 4.

6 Conclusions

In this paper was highlighted the basic foundations involved with the fuzzy inference system tuning process from unconstrained optimization techniques. To validation purposes, the results obtained by the proposed approach were compared to those provided from the ANFIS. Based on these promising results, new perspectives of researches related to the tuning process of fuzzy inference systems can be investigated.

References

1. Guillaume, S.: Designing Fuzzy Inference Systems from Data: An Interpretability-Oriented Review. *IEEE Trans. Fuzzy Systems.* 9, 426–443 (2001)
2. Casillas, J., Cordon, O., del Jesus, M.J., Herrera, F.: Genetic Tuning of Fuzzy Rule Deep Structures Preserving Interpretability and Its Interaction with Fuzzy Rule Set Reduction. *IEEE Trans. on Fuzzy Systems* 13, 13–29 (2005)
3. Pal, K., Mudi, R.K., Pal, N.R.: A New Scheme for Fuzzy Rule-Based System Identification and Its Application to Self-Tuning Fuzzy Controllers. *IEEE Trans. on Systems, Man and Cybernetics (Part B)* 32, 470–482 (2004)
4. Dai, X., Li, C.K., Rad, A.B.: An Approach to Tune Fuzzy Controllers Based on Reinforcement Learning for Autonomous Vehicle Control. *IEEE Trans. on Intelligent Transportation Systems* 6, 285–293 (2005)
5. Tung, W.L., Quek, C.: Falcon: Neural Fuzzy Control and Decision Systems Using FKP and PFKP Clustering Algorithms. *IEEE Trans. on Systems, Man and Cybernetics (Part B)* 34, 686–695 (2004)

6. Wan, W., Hirasawa, K., Hu, J., Murata, J.: Relation Between Weight Initialization of Neural Networks and Pruning Algorithms: Case Study on Mackey-Glass Time Series. In: International Joint Conference on Neural Networks, vol. 3, pp. 1750–1755 (2001)
7. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmont (1999)
8. Marquardt, D.: An Algorithm for Least Squares Estimation of Nonlinear Parameters. *J. Soc. Ind. Appl. Math.* 11, 431–441 (1963)
9. Mackey, M.C., Glass, L.: Oscillation and Chaos in Physiological Control Sciences. *Science* 197, 287–289 (1977)
10. Ramot, D., Friedman, M., Langholz, G., Kandel, A.: Complex Fuzzy Logic. *IEEE Trans. on Fuzzy Sets.* 11, 450–461 (2003)
11. Sugeno, M., Yasukawa, T.: A Fuzzy-logic-based Approach to Qualitative Modeling. *IEEE Trans. on Fuzzy Systems* 1, 7–31 (1993)
12. Jang, J.R.: ANFIS - Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. on Systems, Man, and Cybernetics* 23, 665–685 (1993)
13. Mandani, E.H., Assilian, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies* 7, 1–13 (1975)
14. Becker, S.: Unsupervised Learning Procedures for Neural Networks. *International Journal of Neural Systems* 2, 17–33 (1991)
15. Takagi, T., Sugeno, M.: Fuzzy Identification of System and Its Application to Modeling and Control. *IEEE Trans. on Systems, Man, and Cybernetics* 15, 116–132 (1985)
16. Sugeno, M., Kang, G.T.: Structure Identification of Fuzzy Model. *Fuzzy Sets and Systems* 28, 15–33 (1988)
17. Panella, M., Gallo, A.S.: An Input-output Clustering Approach to the Synthesis of ANFIS Networks. *IEEE Trans. on Fuzzy Systems* 13, 69–81 (2005)
18. Huang, G.-B., Babri, H.A.: Universal Approximation Using Incremental Networks with Random Hidden Computation Nodes. *IEEE Trans. on Neural Networks* 17, 879–892 (2006)
19. Li, W., Hori, Y.: An Algorithm for Extracting Fuzzy Rules Based on RBF Neural Network. *IEEE Trans. on Industrial Electronics* 53, 1269–1276 (2006)
20. Kamimura, R., Takagi, T., Nakanishi, S.: Improving Generalization Performance by Information Minimization. *IEEE World Congress on Computational Intelligence* 1, 143–148 (1994)

Neuro-fuzzy Rough Classifier Ensemble*

Marcin Korytkowski^{1,2}, Robert Nowicki^{1,3}, and Rafał Scherer^{1,3}

¹ Department of Computer Engineering, Częstochowa University of Technology
al. Armii Krajowej 36, 42-200 Częstochowa, Poland
<http://kik.pcz.pl>

² Olsztyn Academy of Computer Science and Management
ul. Artyleryjska 3c, 10-165 Olsztyn, Poland
<http://www.owsiiz.edu.pl/>

³ Department of Artificial Intelligence, Academy of Humanities and Economics in Łódź
ul. Rewolucji 1905 nr 64, Łódź, Poland
<http://www.wshe.lodz.pl>
marcink@kik.pcz.czest.pl, robert.nowicki@kik.pcz.pl,
rafal@ieee.org

Abstract. The paper proposes a new ensemble of neuro-fuzzy rough set classifiers. The ensemble uses fuzzy rules derived by the Adaboost metalearning. The rules are used in an ensemble of neuro-fuzzy rough set systems to gain the ability to work with incomplete data (in terms of missing features). This feature is not common among different machine learning methods like neural networks or fuzzy systems. The systems are combined into the larger ensemble to achieve better accuracy. Simulations on a well-known benchmark showed the ability of the proposed system to perform relatively well.

Keywords: Neuro-fuzzy, rough sets, classifier ensemble.

1 Introduction

By combining various soft computing and machine learning methods we can achieve some kind of their synergy. The proposed solution joins neural networks, fuzzy systems, rough sets and ensemble learning. Neural networks have ability to perfectly fit to data. Fuzzy logic use interpretable knowledge and rough set systems can operate on data with missing features.

One of soft computing applications is classification which consists in assigning an object described by a set of features to a class. The object $x \in \mathbf{X}$ is described by the vector of features $\mathbf{v} \in \mathbf{V}$. Thus we can equate object x class membership with its feature values $\bar{\mathbf{v}} = [\bar{v}_1, \bar{v}_2, \dots, \bar{v}_n]$ class membership. Consequently, we can use interchangeably x or $\bar{\mathbf{v}}$. Let us assume that fuzzy set $A \subseteq \mathbf{V}$ is given as its membership function $\mu_A(x) = \mu_A(\bar{\mathbf{v}}) = \mu_A(\bar{v}_1, \bar{v}_2, \dots, \bar{v}_n)$ where $\bar{v}_i \in \mathbf{V}_i$ for $i = 1, \dots, n$. We also define the set of all object x features $Q = \{v_1, v_2, \dots, v_n\}$. There are many methods for

* This work was partly supported by the Foundation for Polish Science (Professorial Grant 2005-2008) and the Polish Ministry of Science and Higher Education (Habilitation Project 2007-2010 Nr N N516 1155 33, Special Research Project 2006-2009, Polish-Singapore Research Project 2008-2010, Research Project 2008-2010).

classifying data. The traditional statistical classification procedures [1] apply Bayesian decision theory and assume knowledge of the posterior probabilities. Unfortunately, in practical situations we have no information about an underlying probability model and Bayes formula cannot be applied. Over the years, numerous classification methods were developed [1] based on neural networks, fuzzy systems [2][3][4][5], support vector machines, rough sets and other soft computing techniques. These methods do not need the information about the probability model. Yet, they usually fail to classify correctly in case of missing data (features). Generally, there are two ways to solve the problem of missing data:

- Imputation - the unknown values are replaced by estimated ones. The estimated value can be set as the mean of known values of the same feature in other instances. An another idea is to apply the nearest neighbor algorithm based on instances with known value of the same feature. The statistical method can be also used.
- Marginalisation - the features with unknown values are ignored. In this way the problem comes down to the classification in lower-dimensional feature space.

Fuzzy classifiers are frequently used thanks to their ability to use knowledge in the form of intelligible IF-THEN fuzzy rules. They fall into one of the following categories [6], depending on the connective between the antecedent and the consequent in fuzzy rules:

- (i) Takagi-Sugeno method - consequents are functions of inputs,
- (ii) Mamdani-type reasoning method - consequents and antecedents are related by the min operator or generally by a t-norm,
- (iii) Logical-type reasoning method - consequents and antecedents are related by fuzzy implications, e.g. binary, Łukasiewicz, Zadeh etc.

Unfortunately neuro-fuzzy systems are not able to cope with missing data. Here rough set systems show their advantage of coping with missing data. They describe the uncertainty of an object classification taking into consideration limited knowledge about the object.

Classifiers can be combined to improve accuracy [7]. By combining intelligent learning systems, the model robustness and accuracy is nearly always improved, comparing to single-model solutions. Popular methods are bagging and boosting which are meta-algorithms for learning different classifiers. They assign weights to learning samples according to their performance on earlier classifiers in the ensemble. Thus subsystems are trained with different datasets created from the base dataset.

In this paper we will combine fuzzy methods with the rough set theory [8][9][10] and classifier ensemble methods. An ensemble of neuro-fuzzy systems is trained with the AdaBoost algorithm and the backpropagation [11]. Then rules from neuro-fuzzy systems constituting the ensemble are used in a neuro-fuzzy rough classifier. In this way, we obtain rules that are perfectly fitted to data and use them in the classifier which can operate on data with missing features.

2 Neuro-fuzzy Rough System for Classification

In the case of classification, the information about an object x membership to the class ω_j can be fuzzy and is specified by the corresponding consequent ($\bar{z}_j^k = \mu_{\omega_j}(x)$). In such a case fuzzy rules have the following form [3]

$$\begin{aligned}
R^k: & \text{ IF } v_1 \text{ is } A_1^k \text{ AND } v_2 \text{ is } A_2^k \text{ AND } \dots \\
& \dots \text{ AND } v_n \text{ is } A_n^k \text{ THEN } x \in \omega_1(\bar{z}_1^k), x \in \omega_2(\bar{z}_2^k), \dots, \\
& \dots, x \in \omega_m(\bar{z}_m^k)
\end{aligned} \tag{1}$$

where observations v_i of the object x are independent variables, $k = 1, \dots, N$, N is the number of rules, and \bar{z}_j^k is the membership degree of the object x to the j -th class ω_j according to rule k . We also assume that memberships of objects to classes are crisp rather than fuzzy, i.e.

$$\bar{z}_j^k = \begin{cases} 1 & \text{if } x \in \omega_j \\ 0 & \text{if } x \notin \omega_j \end{cases}. \tag{2}$$

In case of rough sets it is assumed that the universe of discourse U is subjected to fuzzy partition Φ into fuzzy sets $F_i \subseteq U$ and approximation of the fuzzy set $A \subseteq U$ is performed on basis of this partition [12], [13].

Definition 1 (Fuzzy rough set). *Fuzzy rough set is a pair $(\underline{\Phi}A, \overline{\Phi}A)$, where set $\underline{\Phi}A$ is Φ -lower approximation of the fuzzy set $A \subseteq U$, and set $\overline{\Phi}A$ is its Φ -upper approximation. Membership functions of sets $\underline{\Phi}A$ and $\overline{\Phi}A$ are determined as follows*

$$\mu_{\overline{\Phi}A}(F_i) = \sup_{x \in U} \min(\mu_{F_i}(x), \mu_A(x)), \tag{3}$$

$$\mu_{\underline{\Phi}A}(F_i) = \inf_{x \in U} \max(1 - \mu_{F_i}(x), \mu_A(x)). \tag{4}$$

Therefore, the lower and upper membership grades are expressed by:

$$\bar{z}_{j*} = \frac{\sum_{r=1}^{N_{\bar{z}_j^k}=1} \mu_{A_L^k}(\bar{v})}{\sum_{r=1}^N \mu_{A_L^k}(\bar{v})} \tag{5}$$

and

$$\bar{z}_j^* = \frac{\sum_{r: \bar{z}_j^k=1}^N \mu_{A_U^k}(\bar{v})}{\sum_{r=1}^N \mu_{A_U^k}(\bar{v})}, \tag{6}$$

where A_L^k and A_U^k are defined as follows

$$A_L^k = \begin{cases} A_*^k & \text{if } \bar{z}_j^k = 1 \\ A^{k*} & \text{if } \bar{z}_j^k = 0 \end{cases} \tag{7}$$

and

$$A_U^k = \begin{cases} A^{k*} & \text{if } \bar{z}_j^k = 1 \\ A_*^k & \text{if } \bar{z}_j^k = 0 \end{cases}. \tag{8}$$

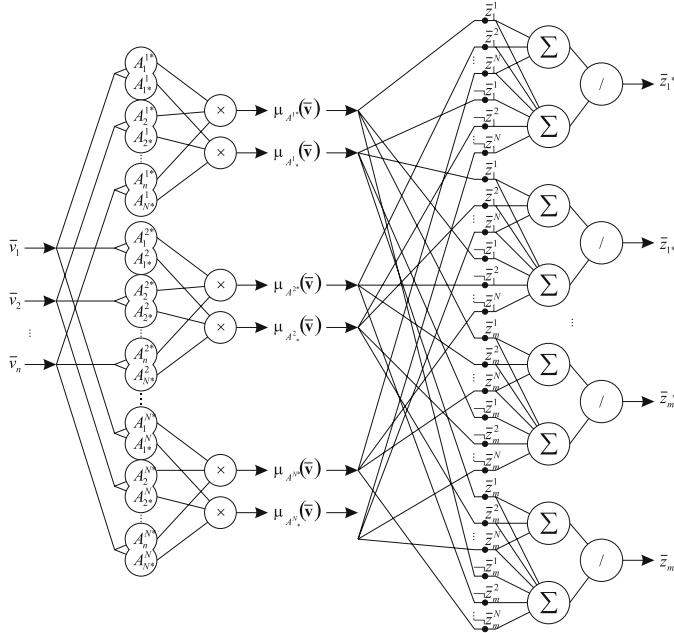


Fig. 1. Neuro-fuzzy rough classifier architecture

The resulting neuro-fuzzy rough architecture is shown in Figure 1. The defuzzification method used in the system is similar to the fuzzy rough classification case. Let \bar{z}_{j*} be the lower membership grade of the object x to the class ω_j and \bar{z}_j^* be its upper membership grade in the form of equations (5) and (6), respectively. In this case, we may fix two numbers (thresholds) z_{IN} and z_{OUT} such that $1 > z_{\text{IN}} \geq z_{\text{OUT}} > 0$. Consequently, the crisp decision can be defined as follows

$$\begin{cases} x \in \omega_j & \text{if } \bar{z}_{j*} \geq z_{\text{IN}} \text{ and } \bar{z}_j^* > z_{\text{IN}} \\ x \notin \omega_j & \text{if } \bar{z}_{j*} < z_{\text{OUT}} \text{ and } \bar{z}_j^* \leq z_{\text{OUT}} \\ \text{perhaps } x \in \omega_j & \text{if } z_{\text{IN}} > \bar{z}_{j*} \geq z_{\text{OUT}} \text{ and } \bar{z}_j^* > z_{\text{IN}} \\ \text{perhaps } x \notin \omega_j & \text{if } \bar{z}_{j*} < z_{\text{OUT}} \text{ and } z_{\text{OUT}} < \bar{z}_j^* \leq z_{\text{IN}} \\ \text{undefined} & \text{otherwise.} \end{cases} \quad (9)$$

When we assume that $z_{\text{IN}} = z_{\text{OUT}} = \frac{1}{2}$, equation (9) takes the following form

$$\begin{cases} x \in \omega_j & \text{if } \bar{z}_{j*} \geq \frac{1}{2} \text{ and } \bar{z}_j^* > \frac{1}{2} \\ x \notin \omega_j & \text{if } \bar{z}_{j*} < \frac{1}{2} \text{ and } \bar{z}_j^* \leq \frac{1}{2} \\ \text{undefined} & \text{otherwise.} \end{cases} \quad (10)$$

3 AdaBoost Algorithm

To build the ensemble we use the AdaBoost algorithm which is the most popular boosting method [14] [15] [16]. Let us denote the l -th learning vector by $\mathbf{z}^l = [x_1^l, \dots, x_n^l, y^l]$, $l = 1 \dots m$ is the number of a vector in the learning sequence, n is the dimension of input vector \mathbf{x}^l , and y^l is the learning class label. Weights D^l assigned to learning vectors, have to fulfill the following conditions

$$(i) 0 < D^l < 1 , \quad (ii) \sum_{l=1}^m D^l = 1 . \quad (11)$$

The weight D^l is the information how well classifiers were learned in consecutive steps of an algorithm for a given input vector x^l . Vector \mathbf{D} for all input vectors is initialized according to the following equation

$$D_t^l = \frac{1}{m}, \quad \text{for } t = 0, \dots, T , \quad (12)$$

where t is the number of a boosting iteration (and a number of a classifier in the ensemble). Let $\{h_t(\mathbf{x}) : t = 1, \dots, T\}$ denotes a set of hypotheses obtained in consecutive steps t of the algorithm being described. For simplicity we limit our problem to a binary classification (dichotomy) i.e. $y \in \{-1, 1\}$ or $h_t(\mathbf{x}) = \pm 1$. Similarly to learning vectors weights, we assign a weight c_t for every hypothesis, such that

$$(i) \sum_{t=1}^T c_t = 1 , \quad (ii) c_t > 0 . \quad (13)$$

Now in the AdaBoost algorithm we repeat steps 1-4 for $t = 1, \dots, T$:

1. Create hypothesis h_t and train it with a data set with respect to a distribution d_t for input vectors.
2. Compute the classification error ε_t of a trained classifier h_t according to the formula

$$\varepsilon_t = \sum_{l=1}^m D_t^l I(h_t(\mathbf{x}^l) \neq y^l) , \quad (14)$$

where I is the indicator function

$$I(a \neq b) = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases} . \quad (15)$$

If $\varepsilon_t = 0$ or $\varepsilon_t \geq 0.5$, stop the algorithm.

3. Compute the value

$$\alpha_t = 0.5 \ln \frac{1 - \varepsilon_t}{\varepsilon_t} . \quad (16)$$

4. Modify weights for learning vectors according to the formula

$$D_{t+1}(\mathbf{z}^l) = \frac{D_t(\mathbf{z}^l) \exp\{-\alpha_t \mathbf{I}(h_t(\mathbf{x}_l) = y^l)\}}{N_t} , \quad (17)$$

where N_t is a constant such that $\sum_{l=1}^m D_{t+1}(\mathbf{z}^l) = 1$. To compute the overall output of the ensemble of classifiers trained by AdaBoost algorithm the following formula is used

$$f(\mathbf{x}) = \sum_{t=1}^T c_t h_t(\mathbf{x}), \quad (18)$$

where

$$c_t = \frac{\alpha_t}{\sum_{t=1}^T |\alpha_t|} \quad (19)$$

is classifier importance for a given training set. The AdaBoost algorithm is a meta-learning algorithm and does not determine the way of learning for classifiers in the ensemble.

4 Experimental Results

In this section we test the proposed approach using the Glass Identification problem [17]. The goal is to classify 214 instances of glass into window and non-window glass basing on 9 numeric features. We took out 43 instances for a testing set. At first we created an Adaboost ensemble of neuro-fuzzy systems. The ensemble consisted of 4 neuro-fuzzy systems each with 4 rules. The NFS systems were learned by the backpropagation algorithm. The fuzzy rules obtained that way were used to create rough fuzzy classifiers. They were created with the use of different implications. The classification results for learning and testing datasets are presented in Table 1

Table 1. Performance of the proposed neuro-fuzzy rough classifier ensemble with different implications on learning/testing Glass dataset

		Classification accuracy [%]
Kleene-Dienes implication		
learning dataset		96.7
testing dataset		84.4
Mamdani/Lukasiewicz implication		
learning dataset		99.3
testing dataset		90.6
Larsen/Reichenbach implication		
learning dataset		98.7
testing dataset		84.4

5 Conclusions

The ensemble is created using the AdaBoost algorithm. The fuzzy subsystems are learned by gradient learning and initialized by modified FCM clustering algorithm. We obtain fuzzy rules from the ensemble and incorporate this rules to the ensemble of neuro-fuzzy rough systems. The systems work on data with missing features. Simulations on a well known benchmark shows very good ability to classify data. Results are comparable to the best ones from the literature [6].

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience Publication, Hoboken (2000)
2. Jang, R.J.S., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence. Prentice Hall, Upper Saddle River (1997)
3. Kuncheva, L.I.: Fuzzy Classifier Design. Studies in Fuzziness and Soft Computing. Physica-Verlag, Heidelberg (2000)
4. Kuncheva, L.: Foundations of Neuro-Fuzzy Systems. John Wiley, Chichester (1997)
5. Wang, L.X.: Adaptive Fuzzy Systems and Control. PTR Prentice Hall, Englewood Cliffs (1994)
6. Rutkowski, L.: Flexible Neuro-Fuzzy Systems. Kluwer Academic Publishers, Dordrecht (2004)
7. Kuncheva, L.: Combining Pattern Classifiers. John Wiley & Sons, Chichester (2004)
8. Pawlak, Z.: Rough sets. International Journal of Information and Computer Science 11, 341–356 (1982)
9. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer, Dordrecht (1991)
10. Pawlak, Z.: Rough sets, decision algorithms and bayes' theorem. European Journal of Operational Research 136, 181–189 (2002)
11. Korytkowski, M., Rutkowski, L., Scherer, R.: On combining backpropagation with boosting. In: 2006 International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence, Vancouver, BC, Canada, pp. 1274–1277 (2006)
12. Dubois, D., Prade, H.: Putting rough sets and fuzzy sets together. In: Słowiński, R. (ed.) Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, pp. 203–232. Kluwer, Dordrecht (1992)
13. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. Internat. J. General Systems 17, 191–209 (1990)
14. Breiman, L.: Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California (1997)
15. Meir, R., Rätsch, G.: An introduction to boosting and leveraging. In: Mendelson, S., Smola, A.J. (eds.) Advanced Lectures on Machine Learning. LNCS (LNAI), vol. 2600, pp. 118–183. Springer, Heidelberg (2003)
16. Schapire, R.E.: A brief introduction to boosting. In: Conference on Artificial Intelligence, pp. 1401–1406 (1999)
17. Asuncion, A., Newman, D.: Uci machine learning repository (2007),
<http://www.ics.uci.edu/~mlearn/MLRepository.html>

Combining Feature Selection and Local Modelling in the KDD Cup 99 Dataset*

Iago Porto-Díaz, David Martínez-Rego, Amparo Alonso-Betanzos,
and Oscar Fontenla-Romero

Department of Computer Science, University of A Coruña, Spain
`{iporto,ciamparo,ofontenla}@udc.es`

Abstract. In this work, a new approach for intrusion detection in computer networks is introduced. Using the KDD Cup 99 dataset as a benchmark, the proposed method consists of a combination between feature selection methods and a novel local classification method. This classification method –called FVQIT (Frontier Vector Quantization using Information Theory)– uses a modified clustering algorithm to split up the feature space into several local models, in each of which the classification task is performed independently. The method is applied over the KDD Cup 99 dataset, with the objective of improving performance achieved by previous authors. Experimental results obtained indicate the adequacy of the proposed approach.

1 Introduction

Data security is a serious concern in an information society. The overcrowding of internet has contributed to the increases in security risk of information systems. Nowadays, every single company and private individual is exposed to this hazard, which explains why Intrusion Detection Systems (IDS) are now an important part on any network security architecture. IDSs aim to inform the system administrator of any suspicious activities inside the target computer network. In general, there exist two types of Intrusion Detection Systems: Misuse-based and Anomaly-based IDSs [1]. The former try to model abnormal behavior by recording signatures for known attacks, while the latter aspire to model normal behavior in order to be able to detect operations beyond normal practice. Artificial Neural Networks and other Machine Learning techniques have proved to be suitable Anomaly-based IDSs.

Over the last few years, scientific and technological interest in techniques for intrusion detection in computer networks has risen steadily. In 1999, the dataset known as KDD Cup 99 was released in the homonym classifier learning contest [2]. It contains an enormous number of connection records and it has become a valuable benchmark to test intrusion detection systems and, broadly, to test pattern recognition and machine learning algorithms.

* This work was supported in part by Spanish Ministerio de Ciencia e Innovación under Project Code TIN 2006-02402, partially supported by the European Union ERDF, and by Xunta de Galicia under Project Code 08TIC012105PR.

This paper presents a new method for intrusion detection based on a combination between some feature selection methods [3] and a novel classification method [4]. This classification method –called FVQIT (Frontier Vector Quantization using Information Theory)– uses a modified clustering algorithm to split up the feature space into local models, in each of which the classification task is performed independently. Several machine learning techniques have been applied over the KDD Cup 99 dataset. However, and up to the authors' knowledge, this dataset has not been addressed from the perspective of local modeling techniques. An explanation for this could be that these methods might have experienced very poor computational performance when applied to such a large dataset. As will be shown, the FVQIT classification method is able to adequately manage large amounts of data, provided that a feature selection method is applied so as to reduce input dimensionality. The objective in this work is to utilize this feature selection and local modeling combined technique to improve performance achieved over the KDD Cup 99 dataset by previous authors.

The remainder of this work is organized as follows. Sect. 2 details the feature selection methods utilized. Sect. 3 explains the classification method chosen. In Sect. 4 an empirical comparative study over the application of the proposed method to the KDD Cup 99 dataset is performed. Lastly, Sect. 5 summarizes the conclusions inferred and proposes some future lines of research.

2 Data Preprocessing

Feature Selection (FS) is a technique that tries to eliminate irrelevant and/or redundant attributes. In this manner, classification algorithms benefit from the decrease in the number of features, improving their predictive accuracy and shortening their learning period. In fact, one of the main problems when facing the KDD Cup 99 dataset is precisely the computational efficiency due to its enormous size.

There is a large number of feature selection methods, which are usually classified into two main models: wrapper methods and filter methods [5]. While the former optimize a predictor to carry out selection, the latter rely on general characteristics of data to get the features selected. As wrappers are more computationally expensive than filter methods, and because of the large size of the KDD Cup 99 dataset, in this work the filter approach is adopted, as in [3], where combinations of discretizers and filters were used so as to improve classification performance over large datasets.

2.1 Filter Methods: INTERACT and Consistency-Based

Among the filters utilized in [3], INTERACT [6] and Consistency-based Filter [7] have been chosen due to their good performance. INTERACT is a new method based on the interaction between features, from an Information Theory point of view; while Consistency-based follows a more classical approach, evaluating consistency between classes.

- *INTERACT*: This algorithm is based on symmetrical uncertainty (SU) [8], which is defined as the ratio between the Information Gain (*IG*) and the Entropy (*H*) of two features, *x* and *y*:

$$SU(x, y) = 2 \frac{IG(x/y)}{H(x) + H(y)} \quad (1)$$

where $IG(x/y) = H(y) + H(x) - H(x, y)$ is the Information Gain, $H(x)$ is the entropy, and $H(x, y)$ is the joint entropy.

Besides SU, INTERACT also deals with the consistency contribution (C-contribution). The C-contribution of a feature is an indicator about how significantly the elimination of that feature will affect consistency. The algorithm utilizes SU and C-contribution to assess whether to remove each feature. It is stated in [6] that INTERACT can thus handle feature interaction, and efficiently select relevant features.

- *Consistency-based Filter*: The Consistency-based Filter judges the value of a subset of features according to the level of consistency of class values depending on training data and tries to minimize the number of features. With this in view, a random subset of features *S* is generated in each iteration, and these data are checked against an inconsistency criterion. If the inconsistency rate is lower than a given threshold, *S* becomes the current minimum. The inconsistency criterion is the key for the success of this algorithm. It specifies a quality measure that determines how much the dimensionality of data can be reduced. If the inconsistency rate of data is smaller than a given threshold, the dimensionally reduced data are acceptable.

2.2 Discretization Methods: PKID and EMD

Some features of the KDD Cup 99 dataset present high imbalance and variability, that is to say, their range is very large but most values are brought together inside a small sub-interval. This situation may cause a malfunction in most classifiers. This problem is softened up by using discretization methods. In substance, the process of discretization involves putting continuous values into groups, by means of a number of discrete intervals. Among the discretization algorithms utilized in [3], PKID (Proportional k-Interval Discretization) [9] and EMD (Entropy Minimization Discretization) [10] have been chosen, because both INTERACT and Consistency-based Filter worked best with them.

- *Proportional k-Interval Discretization (PKID)*: Created by Yang & Webb, the idea behind PKID is seeking an appropriate trade-off between the bias and variance of the probability estimation by adjusting the number and size of discretization intervals to the number of training instances. Thus, a numeric attribute X_i is discretized into \sqrt{n} intervals, each of them containing \sqrt{n} instances, where n is the number of training instances with known values $\{x_1, \dots, x_n\}$ for the attribute X_i . In this way, bias and variance are given equal weight. Furthermore, if n increases, both number and size of intervals increases correspondingly, which means PKID can decrease both bias and variance of the probability estimation.

- *Entropy Minimization Discretization (EMD)*: Proposed by Fayyad & Irani, EMD considers the midpoint between each successive pair of sorted values (x_j, x_{j+1}) as a candidate cut-point. For each candidate cut point, all data left at both sides are discretized into two intervals and the resulting class information entropy is calculated. The cut-point for which the entropy is minimum is selected. The process is repeated recursively, hence selecting the minimum-entropy cut-points. The Minimum Description Length Principle (MDL) [11] is utilized as a stopping criterion.

3 Classification Method: FVQIT

The FVQIT Method is a novel supervised architecture for binary classification based on local modelling and information theory [4]. It is composed of two stages (see Fig. 1): first, a piecewise representation of the frontier between classes is constructed by means of a set of nodes; secondly, given that each of these nodes defines a local model, a hyperplane is constructed and adjusted in each of them by using one-layer neural networks.

3.1 Stage 1: Selection of Local Models

Training data are split up into groups by applying the FVQIT algorithm. The objective is to have a set of nodes placed on the frontier between the two classes, in such a way that each of the nodes will represent a local model. In order to achieve this, both data points and nodes are thought to be two kinds of particles, where each particle has a potential field associated to it. These fields induce repulsive and attractive interactions between particles, depending on the sign of the particles. Among data particles, those which belong to different classes have different signs. In this context, the algorithm minimizes a function of energy that calculates the divergence between the Parzen density estimator of data points $f(x)$ and the estimator of the distribution of the nodes $g(x)$, where $f(x)$ and $g(x)$ are:

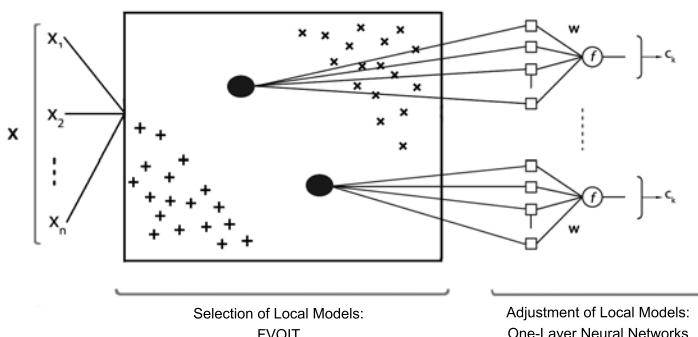


Fig. 1. Structure of the FVQIT Architecture

$$\begin{aligned} f(x) &= \frac{1}{N} \sum_{i=1}^N K\left(x - x_i, \sigma_f^2\right) \\ g(x) &= \frac{1}{N} \sum_{i=1}^N K\left(x - w_i, \sigma_g^2\right) \end{aligned} \quad (2)$$

where N is the number of data points, K is a kernel function, σ_f^2 and σ_g^2 are the variances of the kernel functions, $x_i \in \mathbb{R}^n$ are data points, and $w_i \in \mathbb{R}^n$ are the weights associated to the nodes.

Using the Cauchy-Schwarz inequality and applying logarithms to avoid divisions (see [12]), the function of energy $J(w)$ to be minimized with respect to the location of the nodes can be written as:

$$J(w) = \log \int f^2(x) dx - 2 \log \int f(x) g(x) dx + \log \int g^2(x) dx \quad (3)$$

The first term of (3) ($\log \int f^2(x) dx$) is the cross-correlation among the distributions of data. The second addend ($-2 \log \int f(x) g(x) dx$) is the cross-correlation between the distributions of data and nodes. Finally, the third addend is the cross-correlation among the distributions of nodes. Thus, in order to minimize $J(w)$, the second term needs to be maximized and the third term, minimized. Since data points are stationary, the first term will no longer be considered.

The second term of (3) is rewritten so that the piecewise frontier between classes may be obtained. Two new terms are introduced to replace it. Each one of them is related to each of the two classes, the idea being that the algorithm considers that each node “belongs” to just one of the two classes. So, $J(w)$ is expressed now as:

$$J(w) = 2 \log \int f^+(x) g(x) dx - 2 \log \int f^-(x) g(x) dx + \log \int g^2(x) dx \quad (4)$$

where $f^+(x)$ and $f^-(x)$ are the estimators of the distributions of data of both classes. In order to minimize $J(w)$ in (4), the algorithm tries to: (a) minimize the cross-correlation between distributions of nodes and distributions of data of one class and (b) maximize the cross-correlation between distributions of nodes and distributions of data of the other class. In this way, minimizing the function $J(w)$ will place the nodes along the frontier between both classes. In case of overlapping classes, the corresponding nodes will tend to be situated halfway between classes.

Assuming this formulation, when a minimum of the function of energy $J(w)$ is reached, the nodes are stuck in the frontier between classes and no effective force acts on them. Thus, let $C_+ = \int f^+(x) g(x) dx$ be the contribution of data of one class to $J(w)$; $C_- = \int f^-(x) g(x) dx$ the contribution of the data of the other class, and $V = \int g(x)^2 dx$ the contribution of the interactions among nodes. Using the gradient descent method, it is demonstrated in [4] that the weight update rule for node k becomes:

$$w_k(n+1) = w_k(n) - \eta \left(\frac{\Delta V}{V} + \frac{\Delta C_+}{C_+} - \frac{\Delta C_-}{C_-} \right) \quad (5)$$

where n denotes the learning epoch, η the learning rate, and Δ indicates “derivative with respect to w_k ”.

Initially, each node is placed on the position of a randomly selected data point. So as to get them smoothly approaching the frontier, both the variances of the kernel functions and the learning rate η are *annealed* (decreased) on each learning epoch.

The idea seems to be similar to the Learning Vector Quantization (LVQ), although the background philosophy is actually different, because LVQ places centroids in the center of gravity of classes while FVQIT places nodes in the frontiers between classes.

3.2 Stage 2: Adjustment of Local Models

Local models are generated and associated with each of the aforementioned nodes. Each local model comprises the closest data to its node according to the euclidean distance. In this manner, the problem is completely filled by the set of local models, as input data are always assigned to a local model, following a distance criterion. Since nodes define the frontier between classes, the goal of each local model is to adjust a classifier for the data points in the region of influence around its associated node. Figure 2 depicts the areas of influence of a set of local models adjusted using a 2D example dataset. As nodes –represented by squared dots in Figure 2– are placed on the frontier between classes, it is guaranteed that local models –represented by three different gray levels– are constructed in close proximity to “critical zones”.

Local modelling algorithms may sometimes suffer from temporal efficiency problems, intrinsically associated with the simultaneous training process of several local classifiers. That is why choosing a lightweight algorithm has been a high priority. Thus, the classifier adopted is the One-Layer Neural Network, trained with the efficient algorithm presented in [13]. This algorithm is powerful enough to construct an accurate classifying hyperplane for each region of the problem. Besides, it is very efficient: the task of obtaining the weights of a neuron is performed with a complexity of $O((I + 1)^2)$, where I is the dimension

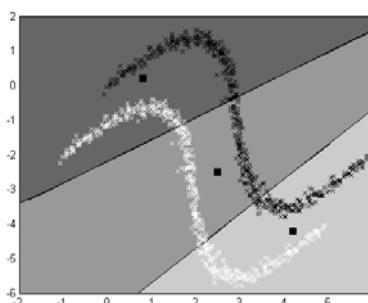


Fig. 2. Areas of influence of a set of three local models

of the input vectors. In this way, training a large number of local models is not so costly. The algorithm is based on an alternative cost function to the classical Mean Squared Error (MSE). This cost function measures the error committed by the network before the non-linear output function is applied. In this manner, the set of weights can be obtained simply by solving a system of linear equations, which requires fewer computational resources than classical methods, as stated in [13]. Finally, after the training process, when a new pattern needs to be classified, the method works as follows: (1) Calculate the closest node w_k to x_n using the Euclidean Distance and (2) Classify x_n using the neural network of the local model associated to w_k .

4 Experimental Study over the KDD Cup 99 Dataset

The KDD Cup 99 dataset is a processed version of the DARPA 1998 dataset [14], which was constructed from a simulation performed by the Defense Advanced Research Projects Agency (DARPA) through the Intrusion Detection Evaluation Program (IDEP) in 1998. It was released for a classifier learning contest, which task was to distinguish between legitimate and illegitimate connections in a computer network [2], at the KDD (Knowledge Discovery and Data Mining) Conference in 1999. The training dataset consists of about 5,000,000 connection records (even though a reduced training dataset containing 494,021 records was released as well) [15]. Each record contains values of 41 variables which describe different aspects of the connection, and the value of the class label (either normal, either the specific attack type). The test dataset comprises 311,029 records and has certain peculiarities: (a) test data is not from the same probability distribution as training data, and (b) the test dataset includes specific attack types not included in the training dataset.

Following the KDD Cup contest, the dataset has been extensively used for development of intrusion detection systems. Specifically, it is utilized as a benchmark for pattern recognition and machine learning algorithms. The dataset is very demanding not only because of its size but also due to the great inner variability among features. For those reasons, the KDD Cup 99 dataset is one of the most challenging classification problems nowadays. Despite that KDD Cup 99 is a multiclass dataset, it can be treated as a binary dataset, simply by grouping all specific attack types into one generic *Attack* class. This approach is interesting in the sense that, most of the time, it is enough to distinguish between normal connections and attacks. This transformation has been carried out by other authors [16,17], and there exist several results in the literature which are utilized as part of the comparative study. The percentages of distribution of both attack and normal classes are, respectively, 80.31% and 19.69% in the training dataset, and 80.52% and 19.48% in the test dataset.

In this work, the experimental study performed involves applying the proposed method –FS+FVQIT– to the binary version of the KDD Cup 99 dataset. The aforementioned discretization methods (PKID and EMD) are considered in combination with the above-named filters (INTERACT and Consistency-based).

Table 1. Results obtained by the four versions of the proposed method and by other authors

Method	Error	TP	FP	Feature No.
PKID+Cons+FVQIT	5.95	92.73	0.48	6
EMD+INT+FVQIT	5.40	93.50	0.85	7
EMD+Cons+FVQIT	4.73	94.50	1.54	7
PKID+INT+FVQIT	5.68	93.61	2.75	7
KDD Winner	6.70	91.80	0.55	41
PKID+Cons+C4.5	5.14	94.08	1.92	6
EMD+INT+C4.5	6.69	91.81	0.49	7
5FNs_poly	6.48	92.45	0.86	41
5FNs_fourier	6.69	92.72	0.75	41
5FNs_exp	6.70	92.75	0.75	41
SVM Linear	6.89	91.83	1.62	41
SVM RBF	6.86	91.83	1.43	41
ANOVA ens.	6.88	91.67	0.90	41
Pocket 2cl.	6.90	91.80	1.52	41

Thus, four combinations of discretizer plus filter are analyzed in order to check which subset of features works best with the FVQIT method.

The model is trained with the KDD Cup 99 training dataset –494,021 samples– and is tested using the standard KDD Cup 99 test dataset of 311,029 samples. Three performance measures –standard for previous authors– are employed:

- *Test Error (TE)*: indicates the overall percentage error rate for both classes –*Normal* and *Attack*–.
- *Attack Detection Rate*: also known as True Positive Rate (TP), it shows the overall percentage of detected attacks.
- *False Positive Rate (FP)*: indicates the proportion of normal patterns classified as attacks.

Hereafter, the results of the proposed method –which parameters have been obtained by means of trial-error experimentation– are compared with those obtained by other authors [23][16][17], as can be seen in Table 1. Specifically, the methods chosen are decision trees (C4.5), functional networks (5FN), Support Vector Machines (SVM) models, ANalysis Of VAriance (ANOVA) (ANOVA ens.) and linear perceptrons (Pocket). Font in boldface indicates best results considering all three measures altogether. Last column specifies the number of features employed.

As can be seen in Table 1, the combination PKID+Cons+FVQIT obtains the best result as it improves the performance obtained by the KDD Cup Winner in all three measures utilized, using a considerably reduced number of features. Moreover, this combination outperforms all other results included in this study. Despite the fact that individual values of Error and True Positive rate for the

combination EMD+Cons+FVQIT are better than those for the abovementioned combination –4.73 versus 5.95 and 94.50 versus 92.73–, it must be noted that the variations in percentage between these amounts are quite small –20% and 2% respectively– in contrast to the variation between the values of False Positive rate –1.54 versus 0.48 (300%)–. On the other hand, Error and True Positive rates for EMD+INT+FVQIT, EMD+Cons+FVQIT, and PKID+INT+FVQIT are good, but unfortunately at the expense of False Positive rates, which appear to be somewhat high for all of them.

5 Conclusions and Future Work

The contribution presented in this paper is a method for detecting intrusions based on the combination of feature selection methods with a local classification method. The system designed is able to detect a large number of intrusions while keeping the false positive rate reasonably low, obtaining better performance than the KDD Cup Winner and other authors. Specifically, the combination PKID+Cons+FVQIT obtains the best result. Furthermore, while previous works had already reduced the necessary number of features, we have achieved an improvement in general performance using the same number of features.

When dealing with the multiclass version of the KDD Cup 99 dataset, pattern recognition and machine learning algorithms usually obtain poor performance, as they usually fail to detect most of the less represented types of attack, mainly because the KDD training and test datasets represent dissimilar probability distributions for these minority categories [18]. Thanks to the good results obtained in this work, future work will involve expanding the proposed method to deal with multiclass problems and applying it to the multiclass version of the KDD Cup 99 dataset.

In addition, dynamic methods for initializing the nodes will be considered in order to be able to deal more efficiently with overlapping classes problems.

References

1. Verwoerd, T., Hunt, R.: Intrusion Detection Techniques and Approaches. *Computer Communications* 25(15), 1356–1365 (2002)
2. Elkan, C.: Results of the KDD 1999 Classifier Learning. *ACM SIGKDD Explorations Newsletter* 1(2), 63–64 (2000)
3. Bolon-Canedo, V., Sanchez-Marcano, N., Alonso-Betanzos, A.: A Combination of Discretization and Filter Methods for Improving Classification Performance in KDD Cup 1999 Dataset. In: Proceedings of the International Joint Conference on Neural Networks, IJCNN (in press, 2009)
4. Martinez-Rego, D., Fontenla-Romero, O., Porto-Díaz, I., Alonso-Betanzos, A.: A New Supervised Local Modelling Classifier Based on Information Theory. In: Proceedings of the International Joint Conference on Neural Networks, IJCNN (in press, 2009)
5. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.: Feature Extraction. In: Foundations and Applications. Springer, Heidelberg (2006)

6. Zhao, Z., Liu, H.: Searching for Interacting Features. In: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI, pp. 1156–1167 (2007)
7. Dash, M., Liu, H.: Consistency-based Search in Feature Selection. *Artificial Intelligence Journal* 151(1-2), 155–176 (2003)
8. Press, W.H., Flannery, B.P., Teutolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. Cambridge University Press, Cambridge (1988)
9. Yang, N., Webb, G.I.: Proportional k-Interval Discretization for Naive-Bayes Classifiers. In: EMCL 2001: Proceedings of the 12th European Conference on Machine Learning, pp. 564–575. Springer, Heidelberg (2001)
10. Fayyad, U.M., Irani, K.B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 1022–1029. Morgan Kaufmann, San Francisco (1993)
11. Grunwald, P.: The Minimum Description Length Principle and Reasoning Under Uncertainty. Unpublished Doctoral Dissertation, University of Amsterdam (1998)
12. Principe, J., Lehn-Schioler, T., Hedge, A., Erdogmus, D.: Vector-Quantization Using Information Theoretic Concepts. *Natural Computing* 4, 39–51 (2005)
13. Castillo, E., Fontenla-Romero, O., Guijarro-Berdiñas, B., Alonso-Betanzos, A.: A global optimum approach for one-layer neural networks. *Neural Computation* 14(6), 1429–1449 (2002)
14. DARPA 1998 Dataset,
<http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/index.html> (cited, March 2009)
15. Levin, I.: KDD 1999 Classifier Learning Contest LLSoft's Results Overview. *ACM SIGKDD Explorations Newsletter* 1(2), 67–75 (2000)
16. Fugate, M., Gattiker, J.R.: Computer Intrusion Detection with Classification and Anomaly Detection, using SVMs. *International Journal of Pattern Recognition and Artificial Intelligence* 17(3), 441–458 (2003)
17. Alonso-Betanzos, A., Sanchez-Marcano, N., Carballal-Fortes, F.M., Suarez-Romero, J., Perez-Sanchez, B.: Classification of Computer Intrusions Using Functional Networks. A Comparative Study. In: ESANN 2007: Proceedings of the European Symposium on Artificial Neural Networks, pp. 25–27 (2007)
18. Sabhnani, M., Serpen, G.: Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set. *Intelligent Data Analysis* 8(4), 403–415 (2004)

An Automatic Parameter Adjustment Method of Pulse Coupled Neural Network for Image Segmentation

Masato Yonekawa and Hiroaki Kurokawa

School of Computer Science, Tokyo University of Technology,
1404-1, Katakura, Hachioji, Tokyo, Japan

Abstract. A Pulse Coupled Neural Network (PCNN) is proposed as a numerical model of cat visual cortex, and it has been applied to the engineering fields especially in an image processing, *e.g.*, segmentation, edge enhancement, and so on. The PCNN model consists of neurons with two kind of inputs, namely feeding input and linking input and they each have a lot of parameters. The Parameters are used to be defined empirically and the optimization of parameters has been known as one of the remaining problem of PCNN. According to the recent studies, parameters in PCNN will be able to be given using parameter learning rule or evolutionary programming. However these methods require teaching images for the learning. In this study, we propose a parameter adjustment method of PCNN for the image segmentation. The proposed method changes the parameters through the iterations of trial of segmentation and the method doesn't require any teaching signal or teaching pattern. The successful results are obtained in the simulations, and we conclude that the proposed method shows good performance for the parameter adjustment of PCNNs.

Keywords: PCNN, image segmentation, parameter adjustment.

1 Introduction

A Pulse Coupled Neural Network (PCNN) is proposed as a model which can show a synchronous dynamics of neurons' activity in the cat visual cortex [1][2][3]. The PCNN model consists of neurons and two kinds of inputs, namely feeding input and linking input with leaky integrators. These inputs lead the discrete time evolution of neurons' internal state and neurons generate a spike output corresponding to the internal state. The linking and feeding inputs are received from neurons' receptive fields which is defined by synaptic weights and directly from the environment.

On the other hand, in engineering point of view, PCNN is considered as a temporal pulse coding system which shows a synchronous pulse dynamics in the network. Using this significant characteristics of PCNN, a lot of applications to the engineering fields are proposed, especially in the field of an image processing, *e.g.*, segmentation, edge enhancement, and so on [4]-[10].

A pattern segmentation is one of the most significant issue in the image processing and an application of the PCNN for pattern segmentation is proposed [4]. To achieve a pattern segmentation, 2D PCNN which has latticed connection is used and one neuron in the network corresponds to one pixel of the image [11]-[19].

To design the PCNN for particular problem such as pattern segmentation, we have to set a lot of number of parameters, *e.g.*, connection weight matrix, time constant and the other coefficients. This means that we have to find a lot of valid parameters of PCNN empirically through the trial and error. Actually, obtaining the valid parameters of PCNN is sometimes difficult because a lot of parameters are defined in the PCNN and the parameters strongly affect the results of its dynamics.

The method of parameter adaptation based on the evaluation of the output of the PCNN in random parameters has been proposed in conventional study [20]. Also, in recent studies, automatically parameter setting algorithm has been proposed using a scheme of evolutionary programming [21]. These parameters and structure adaptation algorithm achieve successful results for the aim of parameter setting. However, these algorithms require teaching pattern to evaluate the output of the PCNN.

In this study, we propose a novel parameter adjustment method of the PCNN for the pattern segmentation. In our proposed method, it doesn't require any teaching patterns or signals to obtain valid parameters of PCNN. We show an implementation of our proposed method to the image segmentation using 2D-PCNN. In the simulation results, we show that the valid parameters of the 2D-PCNN for the pattern segmentation are successfully obtained.

2 The PCNN Model

PCNN is a model of cat visual cortex proposed by Echorn *et.al.*, [3] and a lot of applications in engineering fields have been proposed especially in an image processing [4] [5] [6]. Figure 1 shows a schematic of the neuron in the PCNN. The model consists of the dendrite model and the soma model. In the PCNN, dendrite model forms connections among neurons and input from an environment, and soma model functions as a spike generator.

In general, the PCNN to achieve image processing has two-dimensional structure with lattice connection among neurons and each neuron in the PCNN receives information from each corresponding pixel via feeding input. The two-dimensional PCNN model is mathematically described as follows. The internal state of the neuron N_{ij} , namely membrane potential in biological model, is given by,

$$U_{ij}(t) = F_{ij}(t)(1 + \beta_{ij}L_{ij}(t)). \quad (1)$$

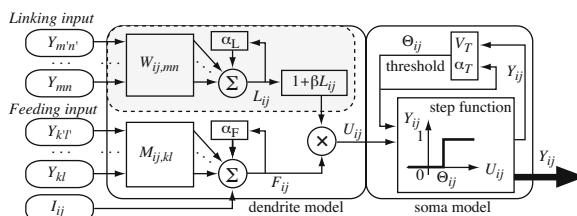


Fig. 1. The schematic of the neuron in Pulse Coupled Neural Network

Note that the indices ij denote neuron number in the two dimensional PCNN. In the PCNN model, there are two kinds of inputs which are propagated via different connections. One is a *feeding input* and the other is a *linking input*. Each input is described as follows, respectively.

$$F_{ij}(t+1) = F_{ij}(t) \exp(-1/\tau_F) + V_F \sum_k \sum_l M_{ij,kl} Y_{kl}(t) + I_{ij}, \quad (2)$$

$$L_{ij}(t+1) = L_{ij}(t) \exp(-1/\tau_L) + V_L \sum_m \sum_n W_{ij,mn} Y_{mn}(t). \quad (3)$$

Where $M_{ij,kl}$ and $W_{ij,mn}$ are weight matrices which define a receptive field of the neuron N_{ij} , I_{ij} is a constant input to the neuron, and $Y_{kl}(t)$ and $Y_{mn}(t)$ are spike output of the neuron N_{kl} and N_{mn} , respectively. This spike output is defined as a step function which is given by,

$$Y_{kl}(t) = \begin{cases} 1 & \text{if } U_{kl}(t) > \Theta_{kl}(t) \\ 0 & \text{else} \end{cases}. \quad (4)$$

Where $\Theta_{kl}(t)$ is a threshold of the action potential of the neuron N_{kl} which is given by,

$$\Theta_{kl}(t+1) = \Theta_{kl}(t) \exp(-1/\tau_T) + V_T Y_{kl}(t) \quad (5)$$

Through Eq.(1)–Eq.(5), parameters, $W_{ij,mn}$, $M_{ij,kl}$, β_{ij} , τ_F , τ_L , τ_T , V_F , V_L , and V_T are decided appropriately.

In this study, the feeding input of the neuron accept only an external input, *i.e.*, pixel intensity, as assumed in conventional studies [5][9]. This assumption is widely used in an application for the image processing. Thus, the feeding input of the neuron is again defined as follows.

$$F_{ij}(t+1) = I_{ij} = \frac{X_{ij}}{255}. \quad (6)$$

Where X_{ij} denotes 8-bit intensity of the pixel P_{ij} . Therefore, unknown parameters of the PCNN which is used for the image segmentation are β_{ij} , τ_L , τ_T , V_L , V_T and $W_{ij,mn}$.

Through the segmentation process using PCNN, an output image from the PCNN is obtained in every time step, and some of them will be adopted as a segmentation image. Figures 2 and 3 show the examples of an image segmentation using PCNN in the case that the object in the input image is a monochrome and a gradation image, respectively. In both results, (a) shows an input image and the others are appropriately selected output images of the PCNN. According to these results, we can know that the results of the segmentation depend on the parameters in PCNN and tunable algorithm for the parameter setting will be required to obtain proper segmentation results.

3 Automatic Adjustment Method for the Parameters in PCNN

As described in previous section, a lot of parameters have to be defined in PCNN for image segmentation and these parameters are designed empirically in conventional studies. To achieve an automatically parameter setting of PCNN for image segmentation,

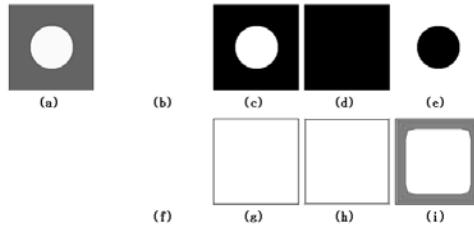


Fig. 2. Results of the segmentation using the PCNN. (a) shows an input image. Figures in upper row, (b)-(e), show the results of segmentation in proper parameters, and figures in lower row, (f)-(i), show the results in wrong parameters.

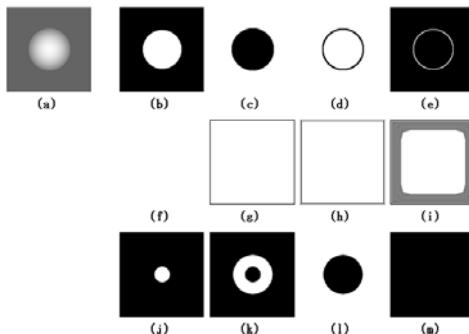


Fig. 3. Another results of the segmentation using the PCNN. (a) shows an input image. Figures in upper row, (b)-(e), show the results of segmentation in proper parameters, figures in middle row, (f)-(i), and lower row, (j)-(m), show the results in wrong parameters.

we propose the parameter adjustment method. In our proposed method, we assumed that only the parameter β_{ij} , τ_T and $W_{ij,mn}$ will be adjusted. Also, in this study, we assumed that the goal of the image segmentation is extracting the plausible area from the input image as an object to be segmented. In other words, the result of the segmentation should represent the characteristics of the object, *i.e.*, a shape and an area.

Some of recent papers had been proposed an algorithm for automatically parameter setting [20][21]. One of these algorithms is based on the evaluation of PCNN's output and reflects its evaluation to the parameters. Also, another algorithm uses evolutionary programming for the parameter adaptation. In any case, an algorithm requires a teaching pattern and the evaluation, which is how similar the output pattern is to the teaching pattern.

On the other hand, our proposed method does not require any teaching pattern or teaching signal. The proposed method is executed through the iteration of trials. Where the trial is defined that the PCNN represent its output pattern for 30 time steps. In our proposed method, through the iteration of the trials, the parameters to be adjusted will be updated gradually.

In our proposed method, the update rules are derived from the characteristics of the parameters. Here, let us summarize the characteristics of parameters τ_T , β_{ij} and $W_{ij,mn}$, as follows.

- In large τ_T , a number of firing neurons at same time step will decrease, and in small τ_T , it will increase.
- In large β_{ij} , firing rate of the neuron N_{ij} will increase and a possibility of the synchronized firing with neighboring neuron increase. On the other hand, in small β_{ij} , activity of neighboring neurons become unrelated each other.
- In large $W_{ij,mn}$, a possibility of the synchronized firing of the connected neurons will increase, *i.e.*, a range of synchronized firing will expand, and in small $W_{ij,mn}$, it will shrink. Here we assume that the $W_{ij,mn}$ is in inverse proportion to the distance between neurons N_{ij} and N_{mn} and a number of connected neurons is initially defined.

Considering these characteristics of the parameters, we propose a parameter adjustment method as described in the following. The basic strategy of the method is to search the appropriate value of parameters which lead the stable V_1 and V_2 to the parameter perturbation. Here $V_1(s)$ and $V_2(s)$ show a variance of the number of firing neurons in a time step and the firing frequency of neurons, respectively. Here, we assumed that the image to be processed has $M \times N$ pixels.

1. Set the initial value of parameters randomly.
2. Execute a trial for 30 time step and calculate following values from the output.
 - S_{1t} : a number of firing neurons at time t
 - S_{2ij} : a number of times that the neuron N_{ij} fires through the trial
3. Derive $M_1(s)$, $M_2(s)$, $V_1(s)$ and $V_2(s)$ as follows. Where s is a number of trials.

$$M_1(s) = \frac{1}{t_{\max}} \sum_t^{t_{\max}} S_{1t}(s), \quad (7)$$

$$M_2(s) = \frac{1}{MN} \sum_i^M \sum_j^N S_{2ij}(s), \quad (8)$$

$$V_1(s) = \frac{1}{t_{\max}} \sum_t^{t_{\max}} (S_{1t}(s) - M_1(s))^2, \quad (9)$$

$$V_2(s) = \frac{1}{MN} \sum_i^M \sum_j^N (S_{2ij}(s) - M_2(s))^2. \quad (10)$$

4. Update the parameters τ_T and β using following equation.

$$\tau_T(s) = \tau_T(s-1) + \tanh \left(\frac{V_1(s)+1}{V_1(s-1)+1} - 1 \right). \quad (11)$$

$$\beta(s) = \beta(s-1) + \tanh \left(\frac{V_2(s)+1}{V_2(s-1)+1} - 1 \right). \quad (12)$$

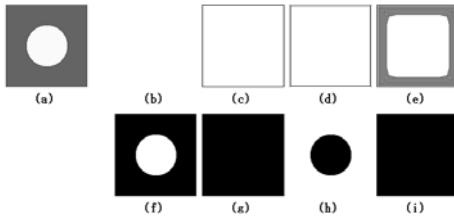


Fig. 4. Results of the segmentation using proposed method (case 1). In first trial ($s = 0$), parameters are defined as, $\tau_T = 1.0$, $\beta = 3.0$, $\tau_L = 1.0$, $V_L = 5.0$ and $V_T = 50$. (a) shows an input image. Figures in upper row show the results of segmentation in initial wrong parameters ($s = 0$), and figures in lower row show the results of segmentation in proper parameters which is obtained by our proposed method ($s = 40$).

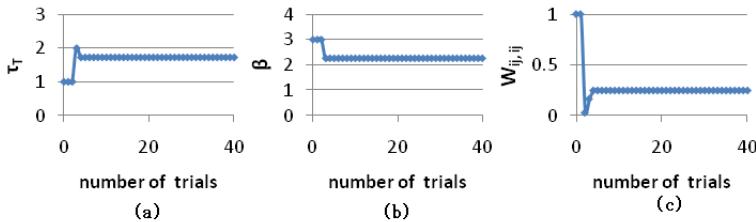


Fig. 5. Changes of parameters by proposed method in case 1: (a) τ_T , (b) β , (c) the linking weight to itself of the neuron $N_{128,64}$, i.e., $W_{128,64,128,64}$

5. Update the parameters $W_{ij,mn}$ using following equation.

$$W_{ij,mn} = \frac{1}{S_{2ij}(s) + 1} \exp \left(-20 \frac{S_{2ij}(s) - K + 1}{P - K + 1} \frac{r}{R} \right), \quad (13)$$

where, r shows the distance between connected neurons N_{ij} and N_{mn} , R shows the maximum distance between connected neurons, P and K show maximum and minimum value of S_{2ij} for all ij , respectively.

6. Return to 2.

4 Simulation Results and Discussions

In this section, we show the simulation results of image segmentation using PCNN, which is described in section 2, with proposed parameter adjustment method. Here the parameters to be adjusted are τ_T , β and $W_{ij,mn}$, and the other parameters are fixed appropriately. We also assume that the neighboring 5×5 neurons are connected using linking input. In the simulations, the initial parameters are wrong parameters which lead the results shown in Fig. 4 and 5, and parameters are adjusted using our proposed method.

In Fig. 4 (a) shows an input image, (b)-(e) show the results in wrong parameters which is also shown in Fig. 5, and (f)-(i) show the results in adjusted parameters using

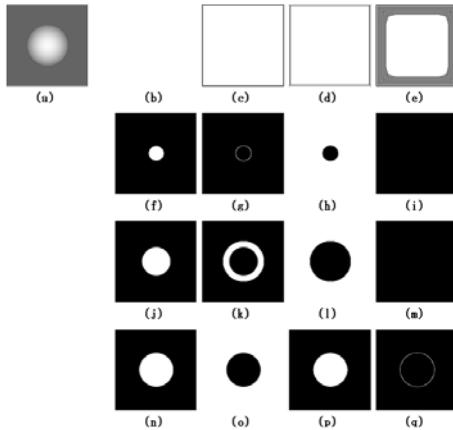


Fig. 6. Results of the segmentation using proposed method (case 2). In first trial ($s = 0$), parameters are defined as, $\tau_T = 1.0$, $\beta = 3.0$, $\tau_L = 1.0$, $V_L = 5.0$ and $V_T = 50$. (a) shows an input image. Figures in upper row show the results of segmentation in initial wrong parameters ($s = 0$). Figures in 2nd and 3rd row show the results that the proposed method is still in progress ($s = 2$, $s = 8$). Figures in lower row show the results of segmentation in proper parameters which is obtained by our proposed method ($s = 40$).

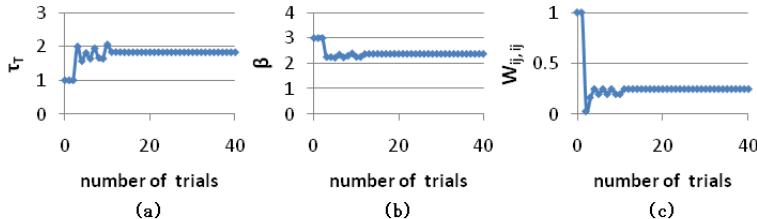


Fig. 7. Changes of parameters by proposed method in case 2: (a) τ_T , (b) β , (c) the linking weight to itself of the neuron $N_{128,64}$, i.e., $W_{128,64,128,64}$

proposed method. Here the results are the selected output images of PCNN in some time steps through the trial and resolution of the image is 256×256 . As shown in Fig 4(f) and (h), the object and the background are segmented in some time steps.

Figure 5(a), (b) and (c) show the convergence characteristics of the parameter τ_T , β and $W_{ij,mn}$ through the adjustment process, respectively. Here, in Fig 5(c), we show the characteristics of the linking weight to itself of the neuron $N_{128,64}$ as a representative value of the distribution of weight matrix. As shown in figures, each parameter is converged to the value to achieve proper segmentation and it is obviously stable after the parameters are converged.

Similar to the simulation results in Fig 4 and 5, we show another simulation results in Fig 6, 7, 8 and 9. In these simulation, initial parameters are set to the wrong parameters which is also shown in Fig 3. Here, the wrong results shown in the middle row in Fig 3 corresponds to the results in Fig 6 and 7, and the lower row in Fig 3 corresponds to the

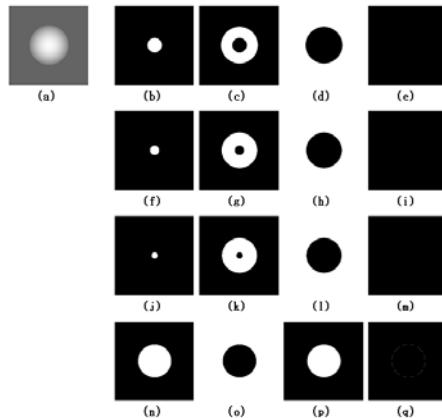


Fig. 8. Results of the segmentation using proposed method (case 3). In first trial ($s = 0$), parameters are defined as, $\tau_T = 2.0$, $\beta = 0.1$, $\tau_L = 1.0$, $V_L = 5.0$ and $V_T = 50$. (a) shows an input image. Figures in upper row show the results of segmentation in initial wrong parameters ($s = 0$). Figures in 2nd and 3rd row show the results that the proposed method is still in progress ($s = 4$, $s = 5$). Figures in lower row show the results of segmentation in proper parameters which is obtained by our proposed method ($s = 40$).

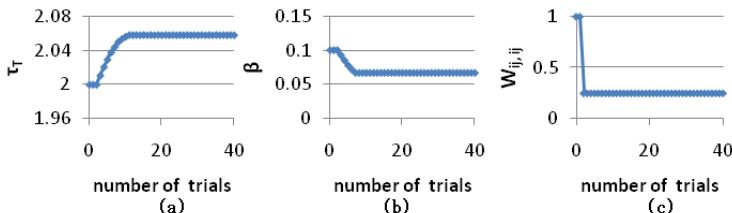


Fig. 9. Changes of parameters by proposed method in case 3: (a) τ_T , (b) β , (c) the linking weight to itself of the neuron $N_{128,64}$, i.e., $W_{128,64,128,64}$

results in Fig. 8 and 9. From these simulations, we also find that the plausible image segmentation is achieved using proposed method and all the parameters are converged. Add to that, in the 2nd row and the 3rd row in Fig. 6 and 8, we also show the results that the proposed method is still in progress. From these results, we can find that the parameters keep changing related to the results of the image segmentation. Through these simulations, we can conclude that our proposed method can detect proper parameters to achieve image segmentation.

Through the trials for the parameter adjustment, the output images of the PCNN approaches to plausible segmentation gradually. Therefore, we show an evaluation of the error which is defined as a ratio of mismatched pixels between an output image and a target image, where the target image of the segmentation is assumed such as in Fig. 10. Each result in Fig. 11 shows the error characteristics of each simulation in Fig. 4, 6 and 8, respectively. Note that, in the simulation in Fig. 11(c), we evaluate the errors of two



Fig. 10. (a) The target object to be segmented (b) Background

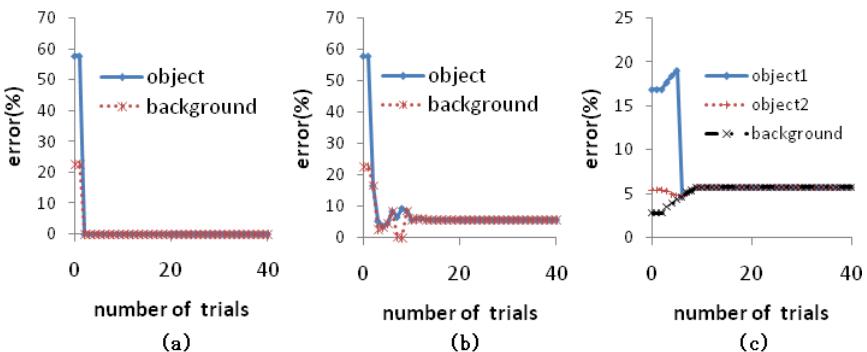


Fig. 11. The characteristics of the error ratio: figures show a difference between object or background obtained by PCNN and the target image shown in Fig. 10

object because two objects are obtained as a segment in early trials in the simulation. From the results, even if the error doesn't converge to zero, the error decreased as trials for parameter adjustment progress and be in stable at a low level.

5 Conclusions

One of the most significant problem of PCNN is its difficulty and complexity of parameter design, especially for the use in engineering fields. In this study, we proposed an automatic parameter adjustment method of PCNN for pattern segmentation. The proposed method adjusts parameters β , τ_T and weight matrices through the iteration of trials. Also one of the most remarkable feature of the proposed method is that the method requires no teaching images or teaching signals for the parameter adjustment. The proposed method was applied to the image segmentation using PCNN and successful results are obtained in the simulations. From the results, we conclude that the proposed method showed a good performance for the parameter adjustment problem. However, an effect of parameters which is not concerned in the proposed method were not revealed in this study. These remaining problem will be discussed in future works.

References

- Engel, A.K., Kreiter, A.K., König, P., Singer, W.: Synchronization of oscillatory neuronal responses between striate and extrastriate visual cortical areas of cat. Proc. Natl. Acad. Sci. USA 88, 6048–6052 (1991)

2. Kruse, W., Eckhorn, R.: Inhibition of sustained gamma oscillations (35–80 Hz) by fast transient responses in cat visual cortex. *Proc. Natl. Acad. Sci. USA* 93, 6112–6117 (1996)
3. Echorn, R., Reitboeck, H.J., Arndt, M., Dicke, P.: Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. *Neural Computation* 2, 293–307 (1990)
4. Echorn, R.: Neural Mechanisms of Scene Segmentation: Recording from the Visual Cortex Suggest Basic Circuits for Liking Field Model. *IEEE Trans. Neural Network* 10(3), 464–479 (1999)
5. Johnson, J.L., Padgett, M.L.: PCNN Models and Applications. *IEEE Trans. Neural Network* 10(3), 480–498 (1999)
6. Lindblad, T., Kinser, J.M.: Image processing using Pulse-Coupled Neural Networks, 2nd edn. Springer, Heidelberg (2005)
7. Ogawa, Y., Ishimura, K., Wada, M.: Stereo Vision using Pulse-Coupled Neural Network. In: Proc. of SICE, pp. 719–724 (2002)
8. Ranganth, H.S., Kuntimad, G.: Image segmentation using pulse coupled neural networks. In: Proc. Int. Conf. Neural Networks, Orlando, FL, vol. 2, pp. 1285–1290 (1994)
9. Ranganth, H.S., Kuntimad, G.: Perfect image segmentation using pulse coupled neural networks. *IEEE Trans. Neural Networks* 10(3), 591–598 (1999)
10. Nazmy, T.M.: Evaluation of the pcnn standard model for image processing purposes. *IJI-CIS* 4(2), 101–111 (2004)
11. Kurokawa, H., Kaneko, S., Yonekawa, M.: A Color Image Segmentation using Inhibitory Connected Pulse Coupled Neural Network. In: Proc. of ICONIP 2008, pp. 399–400 (2008)
12. Gu, X.: A New Approach to Image Authentication using Local Image Icon of Unit-Linking PCNN. In: Proc. of IJCNN, pp. 2015–2020 (2006)
13. Zhou, L., Sun, Y., Zheng, J.: Automated Color Image Edge Detection using Improved PCNN Model. *WSEAS Transactions on Computers* 7(4), 184–189 (2008)
14. Xiong, X., Wang, Y., Zhang, X.: Color Image Segmentation using Pulse-Coupled Neural Network for Locusts Detection. In: Proc. of the International Conference on Data Mining, pp. 410–413 (2006)
15. Sakai, T., Tokito, K., Ogawa, Y., Ishimura, K., Wada, M.: Visual Processing Based on Pulse Coupled Neural Network and Applications. In: Proc of SS 2002, pp. 381–386 (2002)
16. Kinser, J.M.: Foveation by a Pulse–Coupled Neural Network. *IEEE Trans. Neural Networks* 10(3), 621–625 (1999)
17. Becanovic, V., Kermit, M., Eide, A.: Feature extraction from photographic images using a hybrid neural network. In: Proc. Ninth Workshop on Virtual Intelligence/Dynamic Neural Networks, vol. 3728, pp. 351–361. SPIE
18. Ma, Y.-D., Dao, R.-L., Lian, L., Fei, S.: A new algorithm of image segmentation based on pulse–coupled neural networks and entropy of images. In: Proc. Int. Conf. Neural Information Processing (2001)
19. Gu, X.-D., Guo, S.-D., Yu, D.-H.: A new approach for automated image segmentation based on unit-linking PCNN. In: Proc. Int. Conf. Machine Learning and Cybernetics, vol. 1, pp. 175–178 (2002)
20. Szekely, G., Lindblad, T.: Parameter adaptation in a simplified pulse-coupled neural network. In: Proc. Wkshp. Virtual Intell./DYNN, VI-DYNN 1998, Stockholm, Sweden, vol. 3728, SPIE (1998)
21. Berg, H., Olsson, R., Lindblad, T., Chilo, J.: Automaic design of pulse coupled neurons for image segmentation. *Neurocomputing* 71, 1980–1993 (1988)

Pattern Identification by Committee of Potts Perceptrons

Vladimir Kryzhanovsky

Center of Optical Neural Technologies of
Scientific Research Institute for System Analysis of
Russian Academy of Sciences

44/2 Vavilov Street, 119333 Moscow, Russian Federation

Vladimir.Krizhanovsky@gmail.com, iont.niisi@gmail.com

<http://www.niisi.ru/iont/>

Abstract. A method of estimation of the quality of data identification by a parametric perceptron is presented. The method allows one to combine the parametric perceptrons into a committee. It is shown by the example of the Potts perceptrons that the storage capacity of the committee grows linearly with the increase of the number of perceptrons forming the committee. The combination of perceptrons into a committee is useful when given task parameters (image dimension and chromaticity, the number of patterns, distortion level, identification reliability) one perceptron is unable to solve the identification task. The method can be applied in q-ary or binary pattern identification task.

Keywords: Identification, Potts model, storage capacity, committee, perceptron.

1 Introduction

The analysis of parametric fully connected neural networks [1]-[8] has shown that at the present the parametric vector models of the associative memory are the best both with regard to the storage capacity and noise immunity. At the same time their high recognition characteristics of the aforementioned models were not used up to now. The situation changed after the publication of the paper [9] (see also [10], [12]), where the algorithm of mapping of binary patterns into q-valued ones was proposed. It was also shown in [9] that such mapping allows one to use vector neural networks for storing and processing of signals of any type and any dimension. Moreover, the mapping brings to nothing the main difficulty of all the associative memory systems, which is the negative influence of correlations between the patterns. Thus, in the cited work the authors presented an effective and simple approach, showing how the vector models of neural networks can be used and discussed the advantages related to them.

In the papers [13]-[16], a parametric perceptron based on the Potts neurons was presented. The reliability and the processing speed of the Potts perceptron are orders of magnitude higher than the ones of the fully connected neural networks.

However, there are problems requiring either identification of heavily distorted patterns or identification of very large amount of patterns those the single perceptron can't solve them. Here we present a method, which allows one to estimate the quality of the data identification by the parametric perceptron. The quality of the identification is the measure of proximity between an input vector and the stored pattern associated with the input vector as a result of the identification process. The advantage of this method is that it doesn't require any additional calculations (one has to calculate only the states of the output neurons). This method allows one to combine the parametric perceptrons into a committee, where each perceptron is trained on its own ensemble of patterns with cardinal number $m = M/K$ (divide to conquer strategy), where M is the total pattern amount and K is the number of perceptrons. Patterns are split to different sets randomly. To identify an input vector we initialize all perceptrons of the committee by this input vector. Then all the perceptrons give some answers, but which of them is correct? To define the perceptron, which has identified the input vector with the highest precision, one should use the "winner takes all" strategy: the winner perceptron is the perceptron having maximal quality of identification. The winner perceptron number and its output vector uniquely define the stored pattern, whose distortion is the input vector.

It is shown by the example of the Potts perceptrons that the storage capacity of the committee grows linearly with the increase the number of the perceptrons. However, the price to pay if that the required RAM needed for modeling and the number of arithmetic operations increase linearly too.

2 The Perceptron Scheme

At first, let us examine the scheme of the parametric perceptron for the case of a great number of colored randomized patterns (multiparameter vectors). This scheme can be easily generalized on the case of vectors of an arbitrary dimension, in particular on binary vectors, and on strongly correlated patterns.

So, let us have a set of N-dimensional Q-valued patterns $\{\mathbf{X}_\mu\}$:

$$\mathbf{X}_\mu = (\mathbf{x}_{\mu 1}, \mathbf{x}_{\mu 2}, \dots, \mathbf{x}_{\mu N}) \quad (1)$$

where $\mathbf{x}_{\mu i}$ is the unit vector directed along one of the Cartesian axes of Q-dimensional space ($\mu = \overline{1, m}; i = \overline{1, N}$), that is $\mathbf{x}_{\mu i} \in \{\mathbf{e}_k\}^Q$, where $\{\mathbf{e}_k\}^Q$ is the set of basis vectors of the space R^Q . Each pattern \mathbf{X}_μ is in one-to-one correspondence with its key (identifier) that is a unique randomly generated q-valued n-dimensional vector \mathbf{Y}_μ :

$$\mathbf{Y}_\mu = (\mathbf{y}_{\mu 1}, \mathbf{y}_{\mu 2}, \dots, \mathbf{y}_{\mu n}), \mathbf{y}_{\mu i} \in \{\nu_k\}^q \quad (2)$$

where ν_k is the basis vector of the space R^q ($\nu_k \in R^q$). Here n and q are the free parameters of the problem, which obey the inequality:

$$q^n \geq m \quad (3)$$

There are two strategies of choosing of the free parameters n and q :

1. minimization of the required RAM and the number of the arithmetic operations: q should be small and $n = \log_q m + 1$;
2. maximization of the storage capacity: $n = 2$ and $q = \sqrt{m}$.

For simplicity we suppose that the patterns $\{\mathbf{X}_\mu\}$ are the set of colored images with a number of pixels N and the number of different colors Q . In this case with each color we associate an integer number ranging between 1 and Q , or, equivalently, a unit vector in Q -dimensional space.

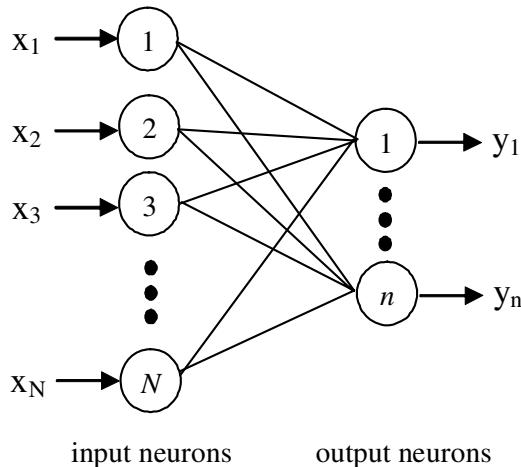


Fig. 1. The scheme of the parametric perceptron

Fig.1 shows the neural network, which is able to retrieve the key corresponding to the pattern, whose distorted image was presented to the network. It consists of two layers of vector neurons, where each neuron of the input layer is connected with all the neurons of the output layer. The algorithm works as follows. We construct the vector neural network based on the standard patterns $\{\mathbf{X}_\mu\}$ and $\{\mathbf{Y}_\mu\}$ (Fig.1). The network consists of N input and n output neurons. To construct the network, we use the parametrical vector neuron approach [6]-[8]. Each neuron of the input and of the output layers may be, respectively, in one of Q and q different states, where $Q \geq 2$ and $q \geq 2$. The k -th state of a neuron of the input layer is associated with the basis vector $\mathbf{e}_k \in R^Q$ of the Q -dimensional space ($k = \overline{1, Q}$). The states of the output neurons are associated similarly. The same as in [6]-[8], the interconnection matrix elements are given by the generalized Hebb rule:

$$\mathbf{T}_{ij} = \sum_{\mu=1}^m \mathbf{y}_{\mu i} \mathbf{x}_{\mu j}^+ \quad (4)$$

where $\mathbf{x}_{\mu j}^+$ is the vector-row ($1 \leq i \leq n$, $1 \leq j \leq N$). Note, in this approach the interconnections \mathbf{T}_{ij} are not scalars, as they are in the standard Hopfield model, but $q \times Q$ matrices.

3 Identification Algorithm of Single Perceptron

Let an image $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ be input to the network. The local field created by all the neurons from the input layer, which is acting on the i -th output neuron, can be calculated as:

$$\mathbf{h}_i = \mathbf{h}_{i0} + \sum_{j=1}^N \mathbf{T}_{ij} \mathbf{x}_j \quad (5)$$

where

$$\mathbf{h}_{0i} = -\frac{N}{Q} \sum_{\mu=1}^m \mathbf{y}_{\mu i} \quad (6)$$

Under the action of the local field the i -th neuron becomes aligned along the basis vector, whose direction is the most close to that of the local field. The calculation algorithm is as follows:

1. The projections of the local field vector \mathbf{h}_i onto all the basis vectors of the q -dimensional space are to be calculated;
2. The maximal projection is to be found. Let it be the projection onto a basis vector ν_{k_i} and its value is $h_i^{\max} = h_i^{k_i}$ ($k_i = \overline{1, q}$);
3. The value ν_{k_i} is set to the output neuron (the "Winner takes all" strategy): $\mathbf{y}_i = \nu_{k_i}$.

As it was shown in [13]-[16], under this dynamics all the components of the encoding output vector \mathbf{Y}_m are retrieved reliable.

The suggested scheme and the identification algorithm of the Potts perceptron aren't classic (compare with [1]-[5] and [13]). The state of a classic Potts neuron is described by the vector $\mathbf{e}_k - \bar{\mathbf{e}}$, where $\bar{\mathbf{e}} = \frac{1}{Q} \sum_{k=1}^Q \mathbf{e}_k$. The conversion of the Potts's vectors to the basis vectors \mathbf{e}_k allows one to accelerate the identification algorithm by a factor of Q , and the calculation of interconnection matrix by a factor of $Q \times q$. It's easy to show that both models are identical.

4 The Identification Quality and Its Applications

The estimation of the quality of identification is defined by the quantity:

$$\langle h_i^{\max} \rangle = \frac{1}{n} \sum_{i=1}^n h_i^{\max}. \quad (7)$$

To understand the meaning of the quantity $\langle h_i^{\max} \rangle$, let's consider the value h_i^{\max} ($h_i^{\max} = h_i^{k_i}$) and make the following assumptions. Let the input vector \mathbf{X}

be a distorted copy of the r-th pattern $\mathbf{X}_r \in \{\mathbf{X}_\mu\}^m$, in which Nb elements are distorted (b is the level of distortion):

$$\mathbf{X} = (\tilde{\mathbf{x}}_{r1}, \tilde{\mathbf{x}}_{r2}, \dots, \tilde{\mathbf{x}}_{rNb}, \mathbf{x}_{rNb+1}, \dots, \mathbf{x}_{rN}). \quad (8)$$

And let the i-th component of \mathbf{Y}_r be equal $\mathbf{y}_{ri} = \nu_{k_i}$. Then substituting the expressions (1) and (3) into (2), we obtain the projections of the local field \mathbf{h}_i :

$$h_i^{k_i} = N(1 - b - 1/Q) + R^{k_i} \quad (9)$$

and

$$h_i^l = R^l, l = \overline{1, q}, l \neq k_i \quad (10)$$

where the quantity

$$R^k = \sum_{\mu \neq r}^m (\nu_k, \mathbf{y}_{\mu i}) \sum_{j=1}^N [(\mathbf{x}_{\mu j}, \mathbf{x}_j) - 1/Q] \quad (11)$$

is the noise constituent with the average $\langle R^k \rangle = 0$ and some dispersion σ_R^2 , and $N(1 - b)$ is the signal constituent. The value $N(1 - b)$ is the number of the non-distorted components in the input vector \mathbf{X} , that is the quantity $N - \langle h_i^{\max} \rangle \approx Nb$ is statistically the Hamming distance between the input vector \mathbf{X} and the stored pattern \mathbf{X}_r associated with \mathbf{X} by the identification process. Note, this method doesn't require any additional calculations (one needs to calculate only the states of the output neurons).

There is a number of problems, which can be solved using the estimation of the quality of identification $\langle h_i^{\max} \rangle$:

1. Some times one needs to identify the cases, in which the input vector \mathbf{X} was obtained as a result of distortion of a pattern not stored into perceptron. To do it one needs to introduce a threshold t and use the following strategy:

- if $\langle h_i^{\max} \rangle \leq t$, then the input vector is an unknown pattern or it is too much distorted;
- if $\langle h_i^{\max} \rangle > t$, then the pattern with the key \mathbf{Y} was given onto the perceptron and we can estimate the part of the distorted components in \mathbf{X} by the expression

$$b \approx 1 - \frac{\langle h_i^{\max} \rangle}{N} \quad (12)$$

2. Some times one needs to identify data, which were distorted by a shift or/and a rotation. The parametric perceptron can't solve this problem directly. To struggle with these types of noise one can use the following strategy:
 - (a) The input image \mathbf{X} is shifted or/and rotated (this operations should not be done at the first iteration);
 - (b) The obtained image is identified by the perceptron. The result of identification is a pair ($\langle h_i^{\max} \rangle, \mathbf{Y}$). The pairs obtained at each iteration are written into a list.

- (c) The number of iteration of items (a) and (b) depends on time interval in which one needs to make a decision;
- (d) In the list of pairs ($\langle h_i^{\max} \rangle$, \mathbf{Y}) one should choose the pair with the largest $\langle h_i^{\max} \rangle$. The key \mathbf{Y} of the found pair would be the output of the system.

Let the input image \mathbf{X} be the pattern \mathbf{X}_r shifted to the right onto 5 pixels and rotated clockwise onto 5 degrees. Then the pair ($\langle h_i^{\max} \rangle$, \mathbf{Y}), which is the result of the identification of the image \mathbf{X} shifted to the *left* onto 5 pixels and rotated *anticlockwise* onto 5 degrees, statistically has the maximal quality of identification $\langle h_i^{\max} \rangle$.

3. There are problems consisting either in identification of heavily distorted patterns, or in identification of a very large amount of patterns that a single perceptron can't solve. Such a problem can be solved by committee of parametric perceptrons. The main idea consists in the following:

- (a) K perceptrons are trained on their own ensembles of patterns with the cardinal number $m = M/K$, where M is the total amount of patterns. Patterns are split to different sets randomly;
- (b) all perceptrons are initialized by the input vector;
- (c) the states of all output neurons are calculated;
- (d) the number of the perceptron, which has the maximum identification quality is determined (WTA strategy);
- (e) the number of the winner perceptron and the values of its output neurons uniquely define the stored pattern, whose distortion is the input vector.

The characteristics of such a committee are considered in the next section.

5 Characteristics of Committee

To estimate the operational capability of the committee consisting of K perceptrons, let us define its characteristics, namely the reliability of identification and the storage capacity, the computational complexity of the algorithm and the capacity of the required RAM needed for the modeling.

The probability of a faithful identification (the reliability of identification) is defined by the expression

$$P = \Pr \left[\bigcap_{i=1}^n \bigcap_{l \neq k_i}^q \left(h_i^{k_i}(1) > h_i^l(1) \right) \right] \Pr \left[\bigcap_{z \neq 1}^K \bigcap_{i=1}^n \bigcap_{l=1}^q \left(h_i^{k_i}(1) > h_i^l(z) \right) \right], \quad (13)$$

where $h_i^l(z)$ is l -th projection of local field of i -th neuron in z -th perceptron. Here in addition to assumptions we made before expression (9), is suggested that the r -th pattern \mathbf{X}_r , which distortion is the input vector \mathbf{X} , was stored in the first perceptron. The first member in (13) is the probability of a faithful identification by the first perceptron. The second one is probability that the identification quality of the first perceptron higher then other.

The $h_i^l(z)$ can be approximately described by Gaussian distribution. In this case we can estimate the expression (13) by the equation

$$P \simeq \left(1 - \frac{1}{\gamma\sqrt{2\pi}} e^{-\frac{\gamma^2}{2}}\right)^{Kqn}, \quad (14)$$

where γ is the so-called signal-noise ratio:

$$\gamma^2 = K \frac{N(1-b)^2 Q q}{2M}. \quad (15)$$

Here b is a part of the distorted components of the input vector. The maximal number of the standards M_{\max} which can be reliably identified by the committee with the error probability less than a given value P_0 , is defined from (3):

$$M_{\max} = K \frac{N(1-b)^2 Q q}{4 \ln(nqK/P_0)}. \quad (16)$$

This estimation we obtained from conditions $1 - P = P_0$ and $P_0 \rightarrow 0$ are performed, when $\gamma^2/2 \gg \ln Kqn/P_0$. One can see, the number of identified patterns M grows proportionally to the number of the perceptrons of the committee. The storage capacity of one Potts perceptron (expression (13) with $K = 1$) is Qq times larger then that of Hopfield model. The computational complexity of the algorithm is estimated by:

$$C = nqNK. \quad (17)$$

As stated above, the computational complexity of the algorithm based on the classical Potts neurons is equal to $nqQNK$. The capacity of the required RAM needed for the modeling is defined by

$$RAM = 32nqQNK \text{ bit.} \quad (18)$$

As it's shown in [17], setting $\mathbf{h}_{0i} = 0$ in Eq.(5) and clipping (binarizing) the elements of synaptic weights \mathbf{T}_{ij} , the required RAM can be decreased by a factor of 32. Moreover, the storage capacity of such a model (for a single perceptron) in the area $N \approx Q$ is twice as large as that of the Potts model, and in the area $N > Q$ the storage capacity is restricted by

$$M_{clip} = qQ \ln \left(1 + \frac{N(1-b)^2}{2 \ln(nq/P_0)} \right). \quad (19)$$

We can increase the magnitude (15) or decrease the magnitudes (16) and (17) varying the free parameters n and q , which are restricted by the inequality (3). Moreover, as each perceptron can be stored and processed on a separate PC, this can be used to decrease the consumption of RAM and to increase the productivity of the committee as a whole.

Note, the expressions (13)-(17) with $K = 1$ describe the characteristic of one Potts perceptron.

6 Conclusions

Usually real-world problems of identification have stringently fixed parameters such as dimension of the input vector N , number of possible values of the input vector components Q , number of patterns M , part of the distorted components of the input vector and the satisfactory maximal probability of identification error P_0 . Unfortunately, the single parametrical perceptron not always can solve such problems, although it has the record storage capacity and the record reliability of identification (see the expressions (13)-(15) when $K = 1$). The current article describe an approach, which allowed combining the parametric perceptrons into the committee. The committee successfully copes with problems described above.

Let us illustrate the above by following example. Picture 2 represent both experimental and theoretical dependencies of the identification error probability on a quantity of the perceptrons forming committee. As we can see, an attempt to solve the assigned problem (please see the picture for problems parameters) using just one identification perceptron has not been successfully, because error probability makes up to 9%. If the committee consisted of 3 perceptrons, the error probability would drop to 0.04%. The difference between theory and experiment is explained in that the real distribution of $h_i^l(z)$ is binomial, not Gaussian.

The core of the presented algorithm is the method of identification quality estimation of the input vector by the parametrical perceptron. The superiority of the approach described above is that there is no need to conduct any additional calculations whatsoever. The capability of the fast identification quality estimation allows to partially tackle shifting and turning noises, to determine whether

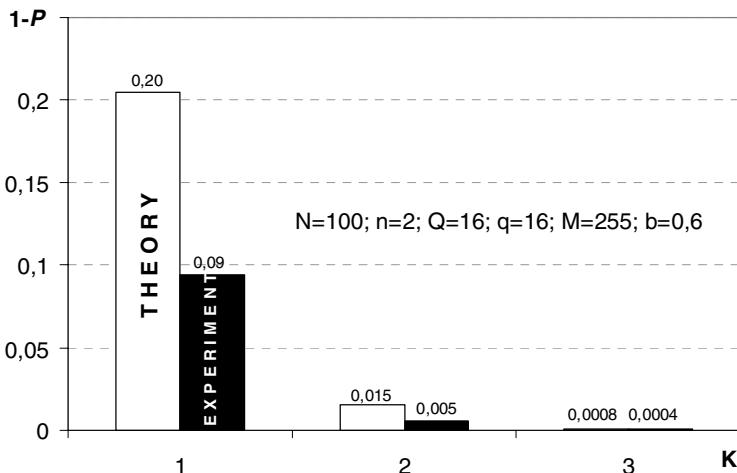


Fig. 2. The theoretical and experimental dependencies of the probability of the identification error $1 - P$ on the number of perceptrons forming the committee. The RAM capacity needed for the committee forming by 1, 2 and 3 perceptrons equals 200Kb, 400Kb and 600Kb, correspondingly.

the input vector is one the stored patterns or not, whether that actual pattern their distortion or not. And most important of it all this capability allows to combine the perceptrons in the committee, which significantly extends the parametrical networks application area. The storage capacity of the committee grows linearly with the increase the number of the perceptrons. The conducted experiments prove this dependence. Of course there are some side effects - the RAM capacity necessary for modulation along with the calculation complexity will lineally increase. Free parameters n , q and K allow solving identification challenges in order of such priorities as reliability, RAM efficiency and capacity. In addition to that any perceptron can be separately kept and processed on different computers, which in fact would lower the RAM load and increase the efficiency of the committee in general.

We used the algorithm presented above to recognize and identify N-dimensional q-valued input vectors. However, rather easily it can be adapted for processing of binary input patterns by means of simple preprocessing, which is the mapping of the binary vector onto q-valued pattern (see [9]-[12]). After that we use our algorithm. Note, the mapping procedure suppresses the correlations between the patterns.

The ideas laid down in this work allow building up perceptron committees, where the core elements would be the following types of neurons:

- Potts neurons (see [1]-[5] and [13]-[16]);
- Phasic vector neurons (see [6]-[8]);
- Phasic vector neurons with clipped synaptic coefficients (see [17]);
- Phaseless vector neurons with clipped synaptic coefficients [18].

The parametric neural networks with clipped synaptic coefficients are of special interest. The clipping of the synaptic coefficients allows decreasing the requirements to RAM necessary for holding of the connection matrix by 32 times. Apart from that, the memory capacity of the clipped phaseless neural network when is twice as high as that of Potts network. All that combined stimulates the demand for such an approach in contemporary systems.

Acknowledgments. I would like to express my special gratitude to Marina Litinskaia and Evgeny Belyaev for help in preparation of the present paper. The work in part supported by the Russian Foundation for Basic Research (#09-07-00159-a).

References

1. Kanter, I.: Potts-glass models of neural networks. *Physical Review A* 37(7), 2739–2742 (1988)
2. Cook, J.: The mean-field theory of a Q-state neural network model. *Journal of Physics A* 22, 2000–2012 (1989)
3. Bolle, D., Dupont, P., van Mourik, J.: Stability properties of Potts neural networks with biased patterns and low loading. *Journal of Physics A* 24, 1065–1081 (1991)

4. Bolle, D., Dupont, P., Huyghebaert, J.: Thermodynamics properties of the q-state Potts-glass neural network. *Phys. Rev. A* 45, 4194–4197 (1992)
5. Wu, F.Y.: The Potts model. *Review of Modern Physics* 54, 235–268 (1982)
6. Kryzhanovsky, B.V., Mikaelyan, A.L.: On the Recognizing Ability of a Neural Network on Neurons with Parametric Transformation of Frequencies. *Doklady Mathematics* 65(2), 286–288 (2002)
7. Kryzhanovsky, B.V., Litinskii, L.B., Fonarev, A.: Parametrical neural network based on the four-wave mixing process. *Nuclear Instruments and Methods in Physics Research A* 502(2-3), 517–519 (2003)
8. Kryzhanovsky, B.V., Litinskii, L.B., Mikaelyan, A.L.: Vector-neuron models of associative memory. In: Proc. of Int. Joint Conference on Neural Networks IJCNN 2004, Budapest, pp. 909–1004 (2004)
9. Kryzhanovsky, B.V., Mikaelyan, A.L.: An associative memory capable of recognizing strongly correlated patterns. *Doklady Mathematics* 67(3), 455–459 (2003)
10. Kryzhanovsky, B.V., Mikaelyan, A.L., Fonarev, A.B.: Vector Neural Net Identifying Many Strongly Distorted and Correlated Patterns. In: Int. conf. on Information Optics and Photonics Technology, Photonics Asia-2004, Beijing-2004. Proc. of SPIE, vol. 5642, pp. 124–133 (2004)
11. Kryzhanovsky, B.V., Kryzhanovsky, V.M., Mikaelian, A.L., Fonarev, A.B.: Parametrical Neural Network for Binary Patterns Identification. *Optical Memory & Neural Network* 14(2), 81–90 (2005)
12. Kryzhanovsky, B.V., Kryzhanovsky, V.M., Fonarev, A.B.: Decorrelating Parametrical Neural Network. In: Proc. of IJCNN Montreal 2005, pp. 1023–1026 (2005)
13. Nadal, J., Rau, A.: Storage capacity of a Potts-perceptron. *J. Phys. I, France* 1, 1109–1121 (1991)
14. Gerlf, F., Krey, U.: Storage capacity and optimal learning of Potts-model perceptrons by a cavity method. *J. Phys. A: Math., Gen.* 27, 7353–7372 (1994)
15. Kryzhanovsky, B.V., Kryzhanovsky, V.M., Magomedov, B.M., Mikaelian, A.L.: Vector Perceptron as Fast search algorithm. *Optical Memory & Neural Network* 13(2), 103–108 (2004)
16. Kryzhanovsky, V.M.: Modified q-state Potts Model with Binarized Synaptic Coefficients. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 72–80. Springer, Heidelberg (2008)
17. Alieva, D.I., Kryzhanovsky, B.V., Kryzhanovsky, V.M.: Vector-neuron models of associative memory with Clipped Synaptic Coefficients. *Pattern Recognition and Image Analysis* (in press)
18. Kryzhanovsky, V., Kryzhanovsky, B., Fonarev, A.: Application of Potts-model Perceptron for Binary Patterns Identification. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 553–561. Springer, Heidelberg (2008)

Is Primal Better Than Dual

Shigeo Abe

Graduate School of Engineering

Kobe University

Rokkodai, Nada, Kobe, Japan

abe@kobe-u.ac.jp

<http://www2.eedept.kobe-u.ac.jp/~abe>

Abstract. Chapelle proposed to train support vector machines (SVMs) in the primal form by Newton's method and discussed its advantages. In this paper we propose training L2 SVMs in the dual form in the similar way that Chapelle proposed. Namely, we solve the quadratic programming problem for the initial working set of training data by Newton's method, delete from the working set the data with negative Lagrange multipliers as well as the data with the associated margins larger than or equal to 1, add to the working set training data with the associated margins less than 1, and repeat training the SVM until the working set does not change. The matrix associated with the dual quadratic form is positive definite while that with the primal quadratic form is positive semi-definite. And the former matrix requires less kernel evaluation. Computer experiments show that for most cases training the SVM by the proposed method is more stable and faster than training the SVM in the primal.

1 Introduction

In a support vector machine (SVM), the input space is mapped into a high-dimensional feature space, and since the mapping function is not explicitly treated by the kernel trick, usually the SVM is trained in the dual form. And many training methods have been developed [123456789]. But because the coefficient vector of the hyperplane is expressed by the kernel expansion, substituting the kernel expansion into the coefficient vector, the SVM in the primal form can be solvable. Based on this idea Chapelle [10] proposed training the SVM in the primal form. Namely, starting from the initial working set selected from the training data, the associated primal quadratic programming program is solved by Newton's method. Then the training data with the associated margins larger than or equal to 1 are deleted from the set, the training data with the margins smaller than 1 are added to the working set, and the above procedure is repeated until no deletion or addition is made to the working set. Chapelle discussed several advantages in training the SVM in the primal over in the dual.

In this paper we propose training SVMs in the dual form in the similar way that Chapelle proposed training SVMs in the primal. Namely, starting from the initial working set, we solve the associated dual quadratic programming problem,

delete from the working set the data whose associated Lagrange multipliers are negative as well as the data with the associated margins larger than or equal to 1, add to the working set the data with the associated margins smaller than 1, and the above procedure is repeated until no data are added to, or deleted from, the working set. We call this method dual SVM and Chapelle's primal SVM. We discuss the advantages of dual SVMs over primal SVM and by computer experiments demonstrate the effectiveness of the dual SVM.

In Section 2, we briefly summarize L1 and L2 SVMs, and in Section 3 we discuss primal SVMs. In Section 4, we compare primal and dual SVMs and clarify the advantages of dual SVMs and in Section 5, we demonstrate the advantages of dual SVMs over primal SVMs for some benchmark data sets.

2 Support Vector Machines

In this section, we summarize support vector machines for two-class problems. In training an SVM, we solve the following optimization problem:

$$\text{minimize} \quad Q(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{p} \sum_{i=1}^M \xi_i^p \quad (1)$$

$$\text{subject to} \quad y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, M, \quad (2)$$

where \mathbf{w} is the weight vector, $\phi(\mathbf{x})$ is the mapping function that maps an m -dimensional input vector \mathbf{x} into the feature space, b is the bias term, (\mathbf{x}_i, y_i) ($i = 1, \dots, M$) are M training input-output pairs, with $y_i = 1$ if \mathbf{x}_i belongs to Class 1, and $y_i = -1$ if Class 2, C is the margin parameter that determines the tradeoff between the maximization of the margin and minimization of the classification error, ξ_i are the nonnegative slack variables for \mathbf{x}_i , and $p = 1$ for an L1 SVM and $p = 2$ for an L2 SVM. We call the value of $y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b)$ the margin for \mathbf{x}_i .

Introducing the Lagrange multipliers α_i , we obtain the following dual problem for the L1 SVM:

$$\text{maximize} \quad Q(\boldsymbol{\alpha}) = \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (3)$$

$$\text{subject to} \quad \sum_{i=1}^M y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \dots, M, \quad (4)$$

and for the L2 SVM:

$$\text{maximize} \quad Q(\boldsymbol{\alpha}) = \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j y_i y_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C} \right) \quad (5)$$

$$\text{subject to} \quad \sum_{i=1}^M y_i \alpha_i = 0, \quad \alpha_i \geq 0 \quad \text{for } i = 1, \dots, M, \quad (6)$$

where $K(\mathbf{x}, \mathbf{x}')$ is a kernel function that is given by $K(\mathbf{x}, \mathbf{x}') = \phi^T(\mathbf{x}) \phi(\mathbf{x}')$ and $\delta_{ij} = 1$ for $i = j$ and 0 for $i \neq j$.

In this study we use polynomial kernels with degree d : $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$ and radial basis function (RBF) kernels: $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$, where d is a positive integer and γ is a positive parameter for slope control.

The KKT complementarity conditions for the L1 SVM are given by

$$\alpha_i \left(\sum_{j=1}^M y_i y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + y_i b - 1 + \xi_i \right) = 0 \quad \text{for } i = 1, \dots, M, \quad (7)$$

$$(C - \alpha_i) \xi_i = 0 \quad \alpha_i \geq 0, \quad \xi_i \geq 0 \quad \text{for } i = 1, \dots, M. \quad (8)$$

For the solution of (3) and (4), if $\alpha_i > 0$, \mathbf{x}_i are called support vectors; especially if $\alpha_i = C$, bounded support vectors and if $0 < \alpha_i < C$, unbounded support vectors.

The KKT complementarity conditions for the L2 SVM are given by

$$\alpha_i \left(\sum_{j=1}^M y_i y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + y_i b - 1 + \frac{\alpha_i}{C} \right) = 0, \quad \alpha_i \geq 0 \quad \text{for } i = 1, \dots, M. \quad (9)$$

Here, $\alpha_i = C \xi_i$.

3 Training Support Vector Machines in the Primal

The optimization problem given by (1) and (2) is converted to the following optimization problem without constraints:

$$\text{minimize} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{p} \sum_{i=1}^M \max(0, 1 - y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b))^p \quad (10)$$

Assume that \mathbf{w} is expressed by

$$\mathbf{w} = \sum_{i=1}^M \beta_i \phi(\mathbf{x}_i), \quad (11)$$

where β_i ($i = 1, \dots, M$) are constants. Substituting (11) into (10), we obtain

$$\begin{aligned} \text{minimize} \quad Q(\beta, b) &= \frac{1}{2} \sum_{i,j=1}^M K(\mathbf{x}_i, \mathbf{x}_j) \beta_i \beta_j \\ &+ \frac{C}{p} \sum_{i=1}^M \max \left(0, 1 - y_i \left(\sum_{j=1}^M \beta_j K(\mathbf{x}_j, \mathbf{x}_i) + b \right) \right)^p. \end{aligned} \quad (12)$$

Let define a set of indices associated with the data that give the optimal solution for the L1 SVM:

$$S = \{i \mid y_i D(\mathbf{x}_i) \leq 1 \text{ for } i \in \{1, \dots, M\}\}, \quad (13)$$

where $D(\mathbf{x})$ is the decision function and is given by $D(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$ and for the L2 SVM

$$S = \{i \mid y_i D(\mathbf{x}_i) < 1 \text{ for } i \in \{1, \dots, M\}\}. \quad (14)$$

Here equality does not hold because of $\alpha_i = C \xi_i$.

We consider whether we can obtain the solution restricting the training data to the data associated with S . Then (12) reduces to

$$\begin{aligned} \text{minimize} \quad Q(\boldsymbol{\beta}, b) = & \frac{1}{2} \sum_{i,j \in S} K(\mathbf{x}_i, \mathbf{x}_j) \beta_i \beta_j \\ & + \frac{C}{p} \sum_{i \in S} \left(1 - y_i \left(\sum_{j \in S} \beta_j K(\mathbf{x}_j, \mathbf{x}_i) + b \right) \right)^p. \end{aligned} \quad (15)$$

For the L1 SVM ($p = 1$), from the KKT conditions (7) and (8), the slack variables ξ_i associated with unbounded support vectors \mathbf{x}_i ($i \in S$) are zero. But in (15), the sum of the slack variables is minimized. Thus, each constraint is not necessarily enforced to zero. In addition, because there is no quadratic term for b , b cannot be determined by this formulation. Therefore, we cannot obtain the solution by solving (15). To solve this problem, Chappell used a Huber loss function, in which a linear loss is combined with a quadratic loss. Because this method gives the approximate solution for the L1 SVM, we do not consider solving L1 SVMs in the primal form.

For the L2 SVM, from (14) ξ_i associated with support vectors \mathbf{x}_i ($i \in S$) are positive. Thus (12) with the restriction of $i \in S$ is equivalent to (15). Let $\boldsymbol{\beta}'_S = (\boldsymbol{\beta}'_S^T, b)^T$, where $\boldsymbol{\beta}_S = \{\beta_i \mid i \in S\}$ and (15) be

$$\text{minimize} \quad Q(\boldsymbol{\beta}'_S) = \mathbf{c}_S^T \boldsymbol{\beta}'_S + \frac{1}{2} \boldsymbol{\beta}'_S^T K_S \boldsymbol{\beta}'_S, \quad (16)$$

where K_S is the $(|S| + 1) \times (|S| + 1)$ matrix, \mathbf{c}_S is the $(|S| + 1)$ -dimensional vector, and

$$\mathbf{c}_{S_i} = -C \sum_{k \in S} y_k K(\mathbf{x}_k, \mathbf{x}_i) \quad \text{for } i \in S, \quad (17)$$

$$\mathbf{c}_{S_b} = -C \sum_{i \in S} y_i, \quad (18)$$

$$K_{S_{ij}} = K(\mathbf{x}_i, \mathbf{x}_j) + C \sum_{k \in S} K(\mathbf{x}_i, \mathbf{x}_k) K(\mathbf{x}_k, \mathbf{x}_j) \quad \text{for } i, j \in S, \quad (19)$$

$$K_{S_{ib}} = C \sum_{j \in S} K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for } i \in S, \quad (20)$$

$$K_{S_{bb}} = C |S|. \quad (21)$$

Solving $\partial Q / \partial \boldsymbol{\beta}'_S = \mathbf{0}$, the optimal solution is given by

$$\boldsymbol{\beta}'_S = -K_S^{-1} \mathbf{c}_S. \quad (22)$$

Here, notice that K_S is positive semi-definite. If K_S is singular, usually a small value is added to the diagonal elements [10]. But because this will increase the number of support vectors, we delete data that cause singularity of the matrix from the working set.

4 Comparison of Training SVMs in the Primal and the Dual

4.1 Training in the Primal

The differences of the training method of the primal SVM and Chapelle's method are that we start training with a small number of chunking data and we avoid singularity of K_S by deleting the associated row and column in decomposing K_S by the Cholesky factorization, instead of adding a small value to the diagonal elements of K_S . We use the variable chunking algorithm. Namely, We solve (22) for the initial working set, delete, from the working set, the data with zero slack variables (the associated margins larger than or equal to 1), add to the working set the data with the positive slack variables (the associated margins smaller than 1), and solve (22), and repeat the above procedure until the same working set is obtained. Let the chunk size be h , where h is a positive integer. Then the procedure is as follows:

1. Set h training data to the working set and go to Step 2.
2. Solve (22) for the working set by the Cholesky factorization. If the diagonal element is smaller than the prescribed value, delete the associated row and column, overwrite the column and row using the next data sample, and resume factorization and obtain β'_S .
3. Delete from the working set the data with zero slack variables, namely \mathbf{x}_i that satisfy $y_i D(\mathbf{x}_i) \geq 1$. And add to the working set at most h most violating data, namely \mathbf{x}_i that satisfy $y_i D(\mathbf{x}_i) < 1$ from the smallest $y_i D(\mathbf{x}_i)$ in order. If the obtained working set is the same with the previous iteration, stop training. Otherwise, go to Step 2.

4.2 Training in the Dual

Similar to the primal SVM, we train the dual SVM. The idea is to eliminate the equality constraint (6) by solving it for one variable and substitute it into (5). Then the problem is reduced to the maximization problem with the positive constraints. We solve the sub-problem without considering the positive constraints, and delete negative variables from the working set. Other procedure is the same with that of the primal SVM.

Consider solving (5) and (6) for the index set S . Solving the equality constraint in (6) for α_s ($s \in S$), we obtain

$$\alpha_s = - \sum_{\substack{i \neq s, \\ i \in S}}^M y_s y_i \alpha_i. \quad (23)$$

Substituting (23) into (6), we obtain the following optimization problem

$$\text{maximize} \quad Q(\boldsymbol{\alpha}_S) = \mathbf{c}_S^T \boldsymbol{\alpha}'_S - \frac{1}{2} {\boldsymbol{\alpha}'_S}^T K_S \boldsymbol{\alpha}'_S \quad (24)$$

$$\text{subject to} \quad \boldsymbol{\alpha}_S \geq \mathbf{0}, \quad (25)$$

where $\boldsymbol{\alpha}_S = \{\alpha_i | i \in S\}$, $\boldsymbol{\alpha}'_S = \{\alpha_i | i \neq s, i \in S\}$, \mathbf{c}_S is the $(|S| - 1)$ -dimensional vector, K_S is the $(|S| - 1) \times (|S| - 1)$ positive definite matrix, and

$$\mathbf{c}_{S_i} = 1 - y_s y_i \quad \text{for } i \neq s, \quad i \in S \quad (26)$$

$$K_{S_{ij}} = y_i y_j \left(K(\mathbf{x}_i, \mathbf{x}_j) - K(\mathbf{x}_i, \mathbf{x}_s) - K(\mathbf{x}_s, \mathbf{x}_j) + K(\mathbf{x}_s, \mathbf{x}_s) + \frac{1 + \delta_{ij}}{C} \right) \quad \text{for } i, j \neq s, \quad i, j \in S. \quad (27)$$

Now the procedure for training the dual SVM is as follows.

1. Set h training data to the working set and go to Step 2.
2. Solve $K_S \boldsymbol{\alpha}'_S = \mathbf{c}_S$ for $\boldsymbol{\alpha}_S$ and using (23) obtain α_s . Determine b by

$$b = y_i - \sum_{j \in S} \alpha_j y_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C} \right) \quad \text{for } i \in S. \quad (28)$$

3. Delete from the working set the data with negative variables, as well as the data \mathbf{x}_i that satisfy $y_i D(\mathbf{x}_i) > 1$. And add to the working set at most h most violating data, namely \mathbf{x}_i that satisfy $y_i D(\mathbf{x}_i) < 1$ from the smallest $y_i D(\mathbf{x}_i)$ in order. If the obtained working set is the same with the previous iteration, stop training. Otherwise, go to Step 2.

Although there may be negative α_i in Step 2, the first equation in (9) is satisfied because $\boldsymbol{\alpha}_S$ is obtained by solving the set of linear equations. Thus any $i (\in S)$ will give the same value. Because we ignore the positive constraints in solving $K_S \boldsymbol{\alpha}'_S = \mathbf{c}_S$ and delete negative variables afterward, the convergence of the above algorithm is not guaranteed.

4.3 Comparison

The differences of the dual SVM and the primal SVM are summarized as follows.

1. Matrix K_S for the dual SVM is positive definite while that for the primal is positive semi-definite. Comparing (17)-(21) and (26), (27), K_s and \mathbf{c}_S for the dual SVM require less kernel evaluations than the primal. Thus, the dual SVM will give more stable solution with less computation time.
2. Mapped support vectors for the primal SVM are interpreted as the independent data that span the empirical feature space [11]. Thus for the linear kernels the number of support vectors for the linear primal SVM is at most the number of the input variables. And any data can be support vectors so long as they span the empirical feature space.

3. Unlike the conventional training methods based on the decomposition technique such as SMO, training for primal and dual SVMs does not guarantee monotonic convergence or may not converge. This is because there is no guarantee that the objective function is monotonic as the iteration proceeds. Thus, to avoid divergence, we may need to incorporate acceleration mechanism such as developed for linear programming SVMs [12].

5 Performance Evaluation

We evaluated performance of the proposed dual SVM with that of the primal SVM using the benchmark data sets shown in Table II which lists the numbers of inputs, classes, training data, and test data. The table also shows the parameter values for L2 SVMs determined by fivefold cross-validation. For instance, $d4$ and $\gamma 10$ mean that the kernels are polynomial kernel with degree 4 and RBF kernels with $\gamma = 10$, and $C10^5$ means that the value of the margin parameter is 10^5 . We used fuzzy one-against-all SVMs [13] and measured the training time using a personal computer (3GHz, 2GB memory, Windows XP operating system). As in [10], we prepared a cache memory with the size equal to the kernel matrix.

Table 1. Benchmark data sets and parameters for L2 SVMs

Data	Inputs	Classes	Train.	Test	Parm.
Thyroid [1]	21	3	3,772	3,428	$d4C10^5$
Blood cell [13]	13	12	3,097	3,100	$\gamma 10C100$
Hiragana-50 [13]	50	39	4,610	4,610	$\gamma 10C1000$
Hiragana-13 [13]	13	38	8,375	8,356	$\gamma 10C500$
Hiragana-105 [13]	105	38	8,375	8,356	$\gamma 10C10^4$
Satimage [14]	36	6	4,435	2,000	$\gamma 200C10$
USPS [15]	256	10	7,291	2,007	$\gamma 10C100$

Table II shows the effect of the chunk size on the performance of primal and dual SVMs for the USPS data set. In the table “Chunk,” “SVs,” “Iterations,” “Kernels,” “Rec.”, and “Time” denote, respectively, the chunk size, the average number of support vectors per one decision function, the number of iterations, the total number of kernel accesses, the recognition rate of the test (training) data set, and training time. “Kernels” means that kernel values are provided through the cache memory if they are in it, and after evaluation if they are not. For “SVs,” “Iterations,” “Kernels,” and “Time” columns, the better value is shown in boldface between dual and primal SVMs.

From the table, the numbers of support vectors for the dual SVM are the same for the five cases, but for the primal SVM, the number increases as the chunk size increases. Although the number of iterations for the primal SVM is smaller, the number of kernel accesses is smaller and training time is shorter for

the dual SVM. This means that the computation burden per iteration for the primal SVM is larger as discussed previously. For the primal SVM, the result did not change even if we added the small value to the diagonal elements.

We examined performance of the case where singularity of the matrix occurs. Table 3 shows the result for the blood cell data set for the linear kernel and $C = 100$. For the primal SVM we also include the results when the diagonal elements are added with 0.00001 denoted as Primal (added). The numeral in the parentheses in the SVs column shows the working set size after training. Thus, for example, for the chunk size of 50, among 497 data only 15 data are support vectors and the remaining data are deleted because of singularity of the matrix.

From the table, training of the dual SVM was the fastest for all the chunk sizes. Comparing the results for the primal SVM with the primal SVM (added), training is faster for the primal SVM (added) but the solutions are different and the number of support vectors increased as the number of the chunk size was

Table 2. Effect of the chunk size for primal and dual SVMs for the USPS data set

Method	Chunk	SVs	Iterations	Kernels	Rec.	Time (s)
Dual	10	597	68	1,889,358,476	95.47 (99.99)	253
	50	597	19	593,529,638	95.47 (100)	114
	100	597	15	585,495,205	95.47 (99.99)	129
	500	597	26	1,926,210,878	95.47 (99.99)	650
	1000	597	26	2,627,126,816	95.47 (99.99)	1527
Primal	10	597	66	28,647,597,296	95.47 (99.99)	1,437
	50	604	16	8,116,273,966	95.47 (99.99)	412
	100	604	11	6,928,602,363	95.42 (99.99)	385
	500	724	7	33,628,904,338	95.47 (99.99)	1,826
	1000	724	7	103,084,629,720	95.47 (99.99)	5,074

Table 3. Comparison of primal and dual SVMs for the blood cell data set with $d = 1$ and $C = 100$

Method	Chunk	SVs	Iterations	Kernels	Rec.	Time (s)
Dual	10	497	55	1,043,872,194	88.97 (91.99)	212
	50	497	17	326,186,052	88.97 (91.99)	71
	100	497	13	274,730,520	88.97 (91.99)	63
	500	497	13	561,872,500	88.97 (91.99)	252
Primal	10	15 (497)	78	110,167,255,737	88.97 (91.99)	4,735
	50	15 (497)	20	27,464,672,230	88.97 (91.99)	1,203
	100	16 (497)	14	19,333,796,617	88.97 (91.99)	848
	500	16 (497)	9	36,858,682,541	88.97 (91.99)	1753
Primal (Added)	10	498	53	53,685,575,039	88.97 (91.99)	2,387
	50	517	14	14,051,320,440	88.97 (91.99)	644
	100	545	9	10,042,016,165	88.90 (91.73)	475
	500	1,073	5	27,336,218,781	87.45 (89.60)	1,287

Table 4. Comparison of primal and dual SVMs for the benchmark data sets

SVM	Data	SVs	Iterations	Kernels	Rec.	Time
Dual	Thyroid	98	1439	2,660,742,540	97.81 (100)	220
	Blood cell	188	13	86,324,885	93.58 (97.19)	10
	H-50	77	20	386,391,968	99.28 (100)	50
	H-13	39	83	2,343,986,783	99.55 (100)	282
	H-105	91	22	812,259,183	100 (100)	147
	Satimage	1001	28	600,106,924	91.70 (99.71)	157
Primal	USPS	597	19	593,529,638	95.47 (100)	114
	Thyroid			No convergence		
	Blood cell	203	10	445,319,582	93.61 (97.19)	21
	H-50	70 (78)	15	605,637,116	99.28 (100)	52
	H-13	99	12	749,712,635	99.70 (99.96)	93
	H-105	111	13	907,360,383	100 (100)	140
Primal	Satiamge	1006	25	26,125,955,619	91.70 (99.71)	1258
	USPS	604	16	8,116,273,966	95.47 (99.99)	412

increased. The results clearly show that the addition of a small positive value to the diagonal elements is not a good strategy for avoiding the singularity of the matrix.

Table 4 lists the results for the primal and dual SVMs using the benchmark data sets. We set the chunk size of 50 for all the cases. For the thyroid data set, training of the dual SVM was very slow. And for the primal SVM the working set size fluctuated considerably and the training did not converge within 10,000 iterations. Except for hiragana-13 and hiragana-105 data sets, the dual SVM was faster and except for the hiragana-50 data set the number of support vectors for the dual SVM was smaller.

From our experiments it is clear that the dual SVM is better than the primal SVM from the standpoints of stable convergence and fast training.

6 Conclusions

In this paper we proposed a new training method for SVMs in the dual form (dual SVMs). Namely, starting with an initial working set, we solve the subproblem expressed in the dual form by Newton's method, delete the data if the associated Lagrange multipliers are non-positive as well as the data with zero slack variables, add the data with positive slack variables, and repeat solving the subproblem until the same working set is obtained. We clarified the advantages of the dual SVM over the primal SVM from the standpoint of positive definiteness of the matrix. By computer experiments we show that training of the dual SVMs was faster than that of the primal SVMs and usually the number of support vectors was smaller. We also showed that adding a small value to the diagonal element for the primal SVMs makes a larger number of support vectors and gives different solutions for the change of the chunk size.

References

1. Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. In: Proc. NNSP, pp. 276–285 (1997)
2. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods: Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1999)
3. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Trans. Neural Networks 11(1), 124–136 (2000)
4. Raicharoen, T., Lursinsap, C.: Critical support vector machine without kernel function. In: Proc. ICONIP 2002, vol. 5, pp. 2532–2536 (2002)
5. Roobaert, D.: DirectSVM: a fast and simple support vector machine perceptron. In: Proc. NNSP, vol. 1, pp. 356–365 (2000)
6. Vishwanathan, S.V.N., Murty, M.N.: SSVM: A simple SVM algorithm. In: Proc. IJCNN 2002, vol. 2, pp. 2393–2398 (2002)
7. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: Advances in Neural Information Processing Systems 13, pp. 409–415. MIT Press, Cambridge (2001)
8. Laskov, P., Gehl, C., Kruger, S., Müller, K.-R.: Incremental support vector learning: Analysis, implementation and applications. Journal of Machine Learning Research 7, 1909–1936 (2006)
9. Abe, S.: Batch support vector training based on exact incremental training. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 295–304. Springer, Heidelberg (2008)
10. Chapelle, O.: Training a support vector machine in the primal. In: Large-Scale Kernel Machines, pp. 29–50. MIT Press, Cambridge (2007)
11. Abe, S.: Sparse least squares support vector training in the reduced empirical feature space. Pattern Analysis and Applications 10(3), 203–214 (2007)
12. Torii, Y., Abe, S.: Decomposition techniques for training linear programming support vector machines. Neurocomputing 72(4–6), 973–984 (2009)
13. Abe, S.: Support Vector Machines for Pattern Classification. Springer, Heidelberg (2005)
14. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007),
<http://www.ics.uci.edu/~mlearn/MLRepository.html>
15. USPS Dataset,
<http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html>

A Fast BMU Search for Support Vector Machine

Wataru Kasai¹, Yutaro Tobe¹, and Osamu Hasegawa²

¹ Department of Computer Intelligence and Systems Science

² Imaging Science and Engineering Laboratory

Tokyo Institute of Technology, Yokohama, 226-8503, Japan

{kasai, ytobe, hasegawa}@isl.titech.ac.jp

Abstract. As described in this paper, we propose a fast learning algorithm of a support vector machine (SVM). Our work is base on the Learning Vector Quantization (LVQ) and we compress the data to perform properly in the context of clustered data margin maximization. For solving the problem faster, we propose a fast Best Matching Unit (BMU) search and introduce it to the Threshold Order-Dependent (TOD) algorithm, which is one of the simplest form of LVQ. Experimental results demonstrate that our method is as accurate as the existing implementation, but it is faster in most situations. We also show the extension of the proposed learning framework for online re-training problem.

Keywords: Kernel Machine, Online re-training, Large data processing.

1 Introduction

The Support Vector Machine (SVM) [1] is a strong tool for many binary or multiple classification tasks. However, SVM's learning process requires much time for training with vast data because it is a quadratic programming problem.

Many researchers have addressed this problem. The Sequential Minimal Optimization (SMO) by Platt [2], which treats only two updated samples called the “active working set” is well known. Some implementations [3] have been adopted. Dong et al. proposed decomposition into some partial problems by particularly addressing a block-diagonal matrix [4]. The Core Vector Machine (CVM) [5] which solves the Minimal Enclosing Ball problem, and its simplified version Ball Vector Machine (BVM) [6] are efficient handful subset finding techniques. Recent works use the cutting plane algorithm (CPA) for performing optimization in linear time. [7]. Its convergence acceleration technique was also proposed [8]. Nevertheless, CPA has only limited application to the linear SVM.

One of the approach which increases performance on such tasks is to reduce the problem size [9][10]. As a reduction technique, Bin Li et al. reported a principled approach in their past work: the Support Cluster Machine (SCM) [11]. They treat neighbor data as a cluster, and show that the whole problem can be formulated by introducing a Probability Product Kernel (PPK). Each cluster is made using a Threshold Order-Dependent (TOD) algorithm [12]. However, their proposition is the integration of the generative model and the discriminative

model. And the cluster they made using the TOD procedure should be rather interpreted as Learning Vector Quantization(LVQ). Based on this point of view, in this paper we show that the variance are not needed for its training. Furthermore, we propose an improved algorithm to build up the compressed data faster than the original algorithm. It is based on TOD, but it includes the fast Best Matching Unit (BMU) search. The improved algorithm retains the accuracy and the online data processing ability that are useful for additional data re-training.

2 LVQ-SVM

The SVM constructs a separating hyperplane via maximization of the margin between the two datasets. Its dual formulation is as described below.

$$\text{maximize : } \sum_{i=1}^N \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} \quad (1)$$

$$\text{subject to : } 0 \leq \alpha_i \leq C, i = 1, \dots, N$$

$$\sum_{i=1}^N y_i \alpha_i = 0 \quad (2)$$

Therein, C is a constant defining the soft-margin. α_i is a Lagrangian corresponding to a training sample $(\mathbf{x}_i, y_i) \in (\mathbf{R}^d, \{-1, 1\})$, and \mathbf{Q} is a matrix whose elements are $Q_{ij} = y_j K(\mathbf{x}_i, \mathbf{x}_j)$, where $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi^T(\mathbf{x}_i) \Phi(\mathbf{x}_j)$.

For a large dataset, the computational cost of the optimization step is expensive for solving Cholesky decomposition of \mathbf{Q} . First, we introduce $\boldsymbol{\theta}$, which is a concentrated neighbor data expression, to reduce the size of matrix \mathbf{Q} . $\boldsymbol{\theta}$ is defined as a single Gaussian to retain the generality of the data expression and to ease calculation.

$$\boldsymbol{\theta}_i = (\boldsymbol{\mu}_i, \Sigma_i, y_i) \quad (3)$$

In that equation, μ_i, Σ_i respectively represent the mean and covariance matrix of the distribution. y_i is a class label of each cluster. The probability of each datum generated from the cluster is

$$p(\mathbf{x}|\boldsymbol{\theta}_i) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)}}{(2\pi)^{\frac{d}{2}} |\Sigma_i|^{\frac{1}{2}}}. \quad (4)$$

We introduce symmetric Kullback–Leibler divergence to obtain a separate hyperplane based on the cluster.

$$\begin{aligned} D(p(\mathbf{x}|\boldsymbol{\theta}_i) || q(\mathbf{x}|\boldsymbol{\theta}_j)) &= \\ \int_{\mathcal{R}^d} p(\mathbf{x}|\boldsymbol{\theta}_i) \log \frac{p(\mathbf{x}|\boldsymbol{\theta}_i)}{q(\mathbf{x}|\boldsymbol{\theta}_j)} dx + \\ \int_{\mathcal{R}^d} q(\mathbf{x}|\boldsymbol{\theta}_j) \log \frac{q(\mathbf{x}|\boldsymbol{\theta}_j)}{p(\mathbf{x}|\boldsymbol{\theta}_i)} dx \end{aligned} \quad (5)$$

These are solvable in the case where p, q is Gaussian.

$$\begin{aligned} & \text{tr}(\Sigma_i \Sigma_j^{-1}) + \text{tr}(\Sigma_j \Sigma_i^{-1}) - 2d \\ & + \text{tr}((\Sigma_i^{-1} + \Sigma_j^{-1})(\mu_i - \mu_j)(\mu_i - \mu_j)^T) \end{aligned} \quad (6)$$

Therein, d is a dimension of the input data. Furthermore, we can introduce the KL-kernel to deal with a nonlinear problem [13].

$$K(p, q) = e^{-aD(p||q)+b} \quad (7)$$

In that equation, a and b are constants. The SCM employed the Probability Product Kernel (PPK) [14]. That expression is as presented below.

$$K(p, q) = \int_{\mathcal{R}^d} p^\rho q^\rho dx \quad (8)$$

If $\rho = 1$ and p and q are Gaussians defined by θ_i, θ_j , then it can be described as shown below.

$$\begin{aligned} K(p, q) &= (2\pi)^{\frac{d}{2}} |\tilde{\Sigma}|^{\frac{1}{2}} |\Sigma_i|^{-\frac{1}{2}} |\Sigma_j|^{-\frac{1}{2}} \\ &\exp(-\frac{1}{2} (\mu_i^T \Sigma_i^{-1} \mu_i + \mu_j^T \Sigma_j^{-1} \mu_j - \tilde{\mu}^T \tilde{\Sigma}^{-1} \tilde{\mu})) \quad (9) \end{aligned}$$

where $\tilde{\Sigma} = (\Sigma_i^{-1} + \Sigma_j^{-1})^{-1}$, $\tilde{\mu} = \Sigma_i^{-1} \mu_i + \Sigma_j^{-1} \mu_j$

For both KL-kernel and PPK, one important common feature exists. If we assume covariance as unit matrix I , then they converge with the well-known Gaussian kernel [1].

$$K(p, q) = e^{-a' \|\mu_i - \mu_j\|^2 + b'}. \quad (10)$$

The SCM suggested that PPK can accommodate $\Sigma \rightarrow 0$ in the testing phase. Nevertheless, when we assume I for all cluster (and that constant variance exists behind testing samples), the cluster expression retaining only average information is interpreted as quantized vector. And it can be processed in the normal SVM with Gaussian kernel. We call this type of learning LVQ-SVM in this paper. Note that this is a natural formulation for the use of the TOD algorithm (described in section 3) because it does not presume data overlap like Gaussian mixture estimation by such as Expectation Maximization (EM) algorithm.

The LVQ-SVM can be optimized using Sequential Minimal Optimization (SMO). It solves the partial problem constructed by two Lagrangians iteratively. It is much faster than the native solution. The next section presents details of our fast LVQ algorithm.

3 A Fast BMU Search in TOD Algorithm

The SCM tried both the TOD algorithm and EM: the former algorithm provides good accuracy and is faster. The TOD is based on competitive learning, we improve it to further fast algorithm.

¹ Note that this is true in the generalized form σI .

3.1 TOD Algorithm

First, we describe the TOD Algorithm.

1. Define the similarity threshold T of a quantized vector by the Euclidean norm. Initialize the quantized vector set S according to the first data.
2. Input new data $\xi \in \mathbf{R}^d$.
3. Search for the best matching unit (BMU) s_1 by $s_1 = \operatorname{argmin}_{s \in S} \|\xi - \mu_s\|$.
4. If the distance between ξ and μ_{s_1} is greater than similarity threshold T , then the input data are a new quantized vector: add it to S and go to Step 2) to process the subsequent data.
5. If the distance is less than T , then update the mean vector of s_1 as (Note that LVQ-SVM can skip the equation (12) for variance)

$$\Delta\mu = \frac{1}{t}(\xi - \mu^{(t-1)}) \quad (11)$$

$$\Delta\Sigma = \frac{(\xi - \mu^{(t)})^T(\xi - \mu^{(t)}) - \Sigma^{(t-1)}}{t}, \quad (12)$$

where t signifies the frequency at which the quantized vector is selected as BMU.

6. Go to Step 2).

The algorithm described above is a simple form of competitive learning; nevertheless, it still has room for improvement of time. We specifically examine the time it takes to search BMU and propose improved algorithm with introduction of the neighbor quantized vector set, which enables realization of the approximate BMU search.

3.2 The Improved BMU Search

1. Define the upper layer threshold as $T^*(> T)$. Initialize the center set of the neighbor quantized vectors $A = \{\omega_1\}$, with ω_1 as equal to the first data. Set the index of the first quantized vector to 1.
2. Input new data $\xi \in \mathbf{R}^d$.
3. Search for the nearest center point as $p = \operatorname{argmin}_{1 \leq i \leq |A|} \|\xi - \omega_i\|$. Next, search BMU only from the quantized vectors whose index is p .
4. If the distance between ξ and μ_{s_1} is less than T , update the center position as

$$\Delta\omega_p = \frac{1}{t \times t^*}(\xi - \mu_{s_1}), \quad (13)$$

where t means the frequency that the quantized vector s_1 is selected as BMU, and t^* is number of quantized vectors whose index is p . Update the quantized vector information by (11) and go to Step 2).

5. If the distance between ξ and μ_{s_1} is greater than T and distance between ξ and ω_p is less than T^* , add a new quantized vector whose position $\mu_{|S|+1} = \xi$ to S , and set its index p . Update ω_p in the following manner.

$$\Delta\omega_p = \frac{1}{t^*}(\xi - \omega_p) \quad (14)$$

Go to Step 2).

6. If the distance between ξ and ω_p is greater than T^* , Add $\omega_{|\Lambda|+1} = \xi$ to Λ . And a new quantized vector whose index is $|\Lambda|$ and position is $\mu_{\text{new}} = \xi$ to S . Go to Step 2).

The possibility exists of generating an overlapped index region in our improved algorithm. However, if that happens, it has little effect on either the recognition performance or calculation time in practice. We call our improved algorithm “iTOD” in this paper. For confirmation, we formally verify that iTOD has lower cost.

Proposition. If the TOD algorithm and iTOD generate the same number of quantized vectors $M >> 1$, and if each set having same index in iTOD includes the same number of vectors, then iTOD’s computational cost is approximately equal to or less than that of the original TOD algorithm.

Proof. Let N be the number of data; let L be the number of indices in our algorithm ($1 \leq L \leq M$). Both the computational costs of TOD and iTOD’s depend practically on the BMU search. Here, the computational cost of the BMU search is $O(NM)$ in TOD, and $O(N(\frac{M}{L} + L))$ in iTOD.

$$M \geq \frac{M}{L} + L \Leftrightarrow 1 \lesssim L \lesssim M \quad (15)$$

We can lay out the fastest situation by setting $L = \sqrt{M}$ on the assumption described above.

4 Experimental Results

We tested the LVQ-SVM and other methods using six large datasets on a 3.0 GHz Intel Xeon Windows XP machine. As special cases, we only cite results of HeroSVM² from their experiment because it is unavailable now. We can review the details later. Kernel and soft-margin parameters are optimized using Grid Search.

4.1 Forest CoverType

The first experiment is on Forest CoverType in the UCI repository³. We select the two major labels “Spruce-Fir” and “Lodgepole Pine” from among all seven

² HeroSVM, MNIST <http://www.cenparmi.concordia.ca/~jdong/HeroSvm.html>

³ Available at <http://kdd.ics.uci.edu/databases/covertype/covertype.html>

Table 1. Result on Forest CoverType: "Data Size" means inputs for optimization step (i.e. Number of quantized vectors or reduced vectors of random setting). The first term in brackets is the LVQ processing time; the second term is the optimization time. Random results are included for purposes of reference.

	LVQ-SVM(iTOD)	LVQ-SVM(TOD)	LibSVM	OCAS ⁴	Random
Data Size	2453	1968	445627	445627	2453
Number of SV	2434	1968	221620	38	973
Accuracy (%)	80.11	79.28	78.58	72.65	60.4
Train-Time (s)	10.15(8.03+2.12)	22.66(21.22 + 1.44)	52494	71	0.67
Test-Time (s)	17.66	14.78	3714	0.15	9.73

labels. The data consist of 495,141 vectors with 54 dimensions. We try ten-fold cross validation with 9:1 train and test split ratio and comparison with the LVQ-SVM using TOD, LibSVM⁴ which is a popular implementation of the SMO Algorithm, OCAS⁵, which solves the convergence accelerated CPA and LibSVM with random reduction preprocessing tuned in to same number of our quantized vectors. The result is presented in Table 1. The first term in brackets shows the LVQ processing time of TOD and iTOD. The second term shows the optimization time. The test time means the sum of 49,514 testing samples. We set $T = 0.5$ for TOD and $T^* = 2.5T$ for iTOD. OCAS is tested using the Octave interface on Vine Linux 4.2 (same PC), and we select the best parameter $C = 0.1$ from among 0.001, 0.01, 0.1, 1.0, and 10. The training time of the proposed method is about 10 seconds, which is the fastest. Random sampling ends in low accuracy. Consequently, it is said that the LVQ process generates a handful of significant vectors for learning. In addition, LVQ-SVM produces a discrimination boundary by fewer support vectors efficiently. Generally speaking, the fewer support vectors leads to a faster classifier in testing phase. LVQ-SVM also presents an advantage for classification time. Although OCAS shows the fewest support vectors because it is a linear solver; OCAS solves the primal problem and provides the sum of linear weights directly. For that reason, the classification time is independent of the number of support vectors and it is fast. CPA appears to be a promising optimization algorithm, and LVQ-SVM can use CPA instead of SMO for the optimization step if CPA deals with a nonlinear problem.

4.2 MNIST

The second experiment is on MNIST² which is a handwritten number (0–9) prediction task. Training data are 60,000 samples; the test data are 10,000 samples. For comparison with Dong's HeroSVM2.1, we use the same feature extraction of 576 dimensions¹⁵. We set $T = 0.5$ and $T^* = 2.5T$. The result is shown in Table 2 (The result of HeroSVM, as obtained on a computer with a 1.7 GHz

⁴ LibSVM <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

⁵ Shogun toolbox <http://www.shogun-toolbox.org/>

Table 2. Result on MNIST

	LVQ-SVM(iTOD)	LVQ-SVM(TOD)	HeroSVM2.1	LibSVM
Data size	15961	16171	60000	60000
Number of SV	3440	3505	24470	9527
Accuracy (%)	99.37	99.38	99.40	99.42
Train Time (s)	101.9(7.42+94.48)	165.8(68.81 + 97.02)	(230)	1682
Test Time (s)	91.56	95.58	-	303

Table 3. KDD Cup 1999, WEB, IJCNN and USPS

	Train	Test	Dim	Class
KDD Cup1999	4898431	311029	127	2
WEB	49749	14951	300	2
IJCNN	49990	91701	22	2
USPS	266079	75383	676	2

Intel Pentium 4 processor). The SCM also tested MNIST database, their results reported was 98.97%.

We also check the LVQ processing time of TOD and improved TOD for changing T (Fig. ①). The property of our improved TOD for changing T^* is shown in Fig. ②. It is said that the proposed method is faster than the original TOD for every T . Additionally, users need not be nervous about decisions of parameter T^* because iTOD converges to TOD even in the worst case. Furthermore, optimal T^* is obtained when the number of indices L approximately equals the square root of the cluster number M .

4.3 KDD Cup 1999, Web, IJCNN, USPS

For comparison of the LVQ-SVM to recent implementations of Core Vector Machine (CVM), which chooses the core set by solving a Minimum Enclosing Ball (MEB), and the Ball Vector Machine (BVM), which substitutes MEB with Enclosing Ball (EB)⁶, we attempt an additional experiment on KDD Cup 1999, Web, IJCNN and USPS⁷. Details of each dataset are shown in Table 3.

In this case, we set $(T, T^*) = (0.01, 10T)$ for IJCNN and set different T for two classes in case of KDD Cup 1999, WEB, and USPS. Each setting is $T_{\text{KDD1}} = 2.0$, $T_{\text{KDD2}} = 1.0$, $T_{\text{WEB1}} = 2.0$, $T_{\text{WEB2}} = 20$, $T_{\text{USPS1}} = 130$ and $T_{\text{USPS2}} = 40$. Actually, T^* is the same in all cases and $2.5T$. The results are presented in Table 4.

Both CVM and BVM present a tradeoff relationship between the learning speed and the recognition rate for parameter ϵ . The details are reported in [6]; we

⁶ Available at <http://www.cse.ust.hk/~ivor/cvm.html>

⁷ KDDCup1999, Web, and IJCNN and USPS are available at <http://www.cse.ust.hk/ivor/cvm.html>

tested $\epsilon = 10^{-5}$ with recommended parameters. LVQ-SVM using iTOD achieves the fastest or second-fastest SVM learning on IJCNN, USPS, and WEB. KDD Cup 1999 is a rare case because the number of support vectors is very small for its data size. Although CVM and BVM present advantages over LVQ-SVM for such special cases, it is said that LVQ-SVM is useful learning framework.

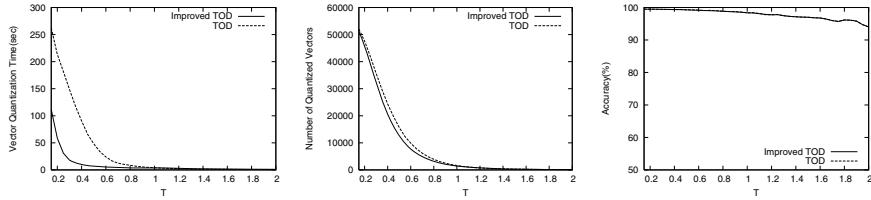


Fig. 1. iTod vs. TOD for changing T on MNIST ($T^* = 2.5T$)

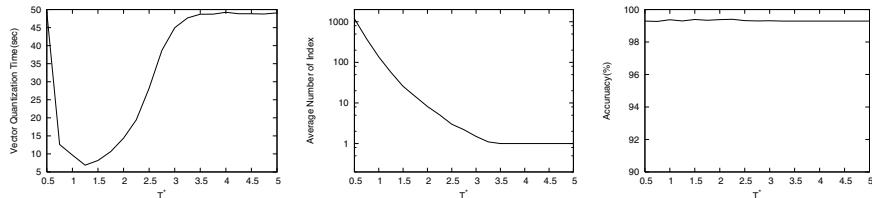


Fig. 2. Details of the iTOD algorithm on MNIST ($T = 0.5$) for changing T^* : The total cluster generation time depends on the number of average (MNIST has 10 classes) indices, and the best condition available when $L \approx \sqrt{M}$ (around 40 in this case). Over-large T^* results in only one index that includes all clusters, and converges to the original TOD. Accuracy is not influenced at all by T^* .

Table 4. Comparison between LVQ-SVM, CVM, and BVM

		LVQ-SVM(iTOD)	LVQ-SVM(TOD)	CVM	BVM
KDD Cup 1999	Number of SV	500	923	53	219
	Accuracy (%)	94.51	94.02	92.34	91.95
	Train Time (s)	50.97(50.77 + 0.20)	130.4(130.1 + 0.34)	1.27	2.09
WEB	Number of SV	3175	3099	9889	3540
	Accuracy (%)	98.91	98.82	99.07	99.04
	Train Time (s)	28.87(23.78 + 5.09)	39.34(34.40 + 4.94)	220.2	23.17
IJCNN	Number of SV	3594	3581	9316	5721
	Accuracy (%)	99.02	99.04	98.37	98.38
	Train Time (s)	20.28(3.08+17.20)	77.48(61.54 + 15.94)	259.4	67.38
USPS	Number of SV	1171	1600	4094	1426
	Accuracy (%)	99.52	99.38	99.50	99.54
	Train Time (s)	33.57(29.30+4.27)	187.94(183.63 + 4.31)	735.8	67.03

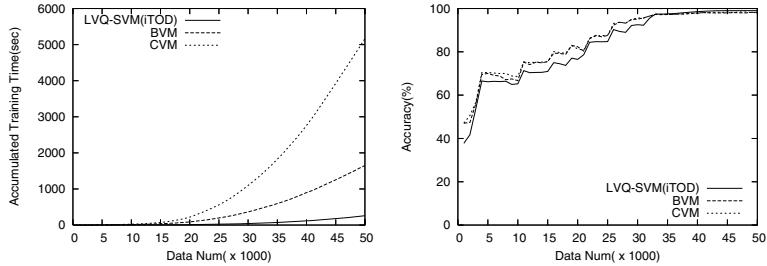


Fig. 3. Performance of online IJCNN data processing: Accumulated training time (left). Accuracy (right).

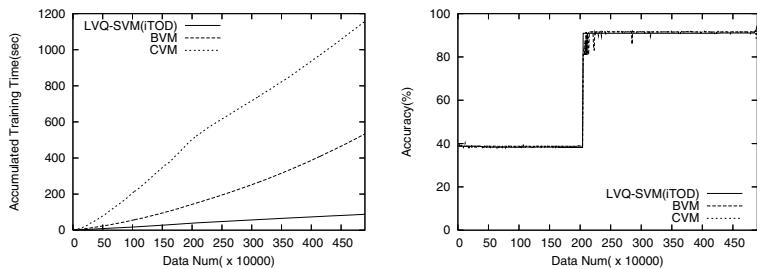


Fig. 4. Performance of online IJCNN data processing: Accumulated training time (left). Accuracy (right).

4.4 Online Re-training

We show that LVQ-SVM is a very efficient online re-training processor. We assume a situation in which data are not obtained all together. In such a case, one might gather data continuously and try re-training of the classifier repeatedly at a certain amount of data collected. We use IJCNN and KDD Cup for this experiment. All classifiers are re-trained once every 1000 samples obtained for IJCNN and every 10,000 samples for KDD Cup. The results are portrayed in Fig. 3 and Fig. 4. The proposed iTOD can accommodate online data, so the accumulated training time is much lower than that of either CVM or BVM, while retaining almost equal accuracy. For the KDD Cup, although influential to accuracy data are coming around 200 step, LVQ-SVM does not miss such data behavior. We briefly presented LVQ based online performance here, and if LVQ algorithm is designed to include forgetting of attributes (e.g. a certain type of SOM) instead of TOD, LVQ-SVM will turn into real-time adaptive SVM.

5 Conclusions

As described herein, we proposed a fast BMU search for Learning Vector Quantization, resulting in a faster SVM learning algorithm. We also showed the advantage of LVQ based approach in treating online re-training problems. we are now

interested in threshold analysis using the point process(e.g. Ripley's k-function) for the future work.

References

1. Cortes, C., Vapnik, V.: Support vector networks. *Machine Learning* 20(3), 273–297 (1995)
2. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: *Advances in Kernel Methods*, pp. 185–208. MIT Press, Cambridge (1999)
3. Fan, R.E., Chen, P.H., Lin, C.J.: Working set selection using second order information for training SVM. *J. Machine Learning Res.* 6, 1889–1918 (2005)
4. Dong, J.X., Krzyzak, A., Suen, C.Y.: Fast SVM training algorithm with decomposition on very large datasets. *IEEE Trans. Pattern Analysis and Machine Intelligence* 27(4), 603–618 (2005)
5. Tsang, I.W., Kwok, J.T., Cheung, P.-M.: Core vector machines: Fast SVM training on very large data sets. *J. Machine Learning Res.* 6, 363–392 (2005)
6. Tsang, I.W., Kocsor, A., Kwok, J.J.T.: Simpler core vector machines with enclosing balls. In: *24th International Conference on Machine Learning*, pp. 911–918 (2007)
7. Joachimes, T.: Training linear svms in linear time. In: *ACM Conference on Knowledge Discovery and Data Mining*, pp. 217–226 (2006)
8. Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for support vector machines. In: *25th International Conference on Machine Learning*, pp. 320–327 (2008)
9. Lee, Y.J., Mangasarian, O.L.: RSVM: Reduced support vector machines. In: *First SIAM International Conference on Data Mining*, pp. 1–17 (2001)
10. Boley, D., Cao, D.: Training support vector machine using adaptive clustering. In: *4th SIAM International Conference on Data Mining*, pp. 126–137 (2004)
11. Li, B., Chi, M., Fan, J., Xue, X.: Support cluster machine. In: *24th International Conference on Machine learning*, pp. 505–512 (2007)
12. Friedman, M., Kandel, A.: *Introduction to pattern recognition: chapter distance functions*, pp. 70–73. Imperical College Press, London (1999)
13. Moreno, P.J., Ho, P.P., Vasconcelos, N.: A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In: *Advances in Neural Information Processing Systems*, vol. 16, pp. 1385–1392 (2003)
14. Jebara, T., Kondor, R., Howard, A.: Probability product kernels. *J. Machine Learning Res.* 5, 819–844 (2004)
15. Dong, J.X., Krzyzak, A., Suen, C.Y.: A fast SVM training algorithm. *International J. Pattern Recognition and Artificial Intelligence* 17(3), 1–18 (2003)

European Option Pricing by Using the Support Vector Regression Approach

Panayiotis C. Andreou¹, Chris Charalambous²,
and Spiros H. Martzoukos²

¹ Durham University, UK

panayiotis.andreou@durham.ac.uk

² University of Cyprus, Cyprus

bachris@ucy.ac.cy, baspiros@ucy.ac.cy

Abstract. We explore the pricing performance of Support Vector Regression for pricing S&P 500 index call options. Support Vector Regression is a novel nonparametric methodology that has been developed in the context of statistical learning theory, and until now it has not been widely used in financial econometric applications. This new method is compared with the Black and Scholes (1973) option pricing model, using standard implied parameters and parameters derived via the Deterministic Volatility Functions approach. The empirical analysis has shown promising results for the Support Vector Regression models.

Keywords: Option pricing, implied volatility, non-parametric methods, support vector regression.

1 Introduction

A call option gives the holder the right, not the obligation, to buy the underlying asset (e.g. a stock) by a certain date (i.e. the expiration date or maturity) by fixing the price of the asset now (i.e. the exercise price). There are American and European styled options. American options can be exercised at any time up to the maturity of the option, whilst European options can be exercised only on the expiration date itself. European styled options can be priced using the Black-Scholes (1973) option pricing model [5]. Moreover, they are generally easier to analyze than American options, and some of the properties of an American option are frequently deduced from those of its European counterpart (see Hull, 2008 [13]).

The Black and Scholes (BS) (1973) model is considered as the most prominent achievement in the option pricing theory. Empirical research has shown that the formula suffers from systematic biases known as the volatility smile/smirk anomaly that result from the simplistic assumptions that governs its pricing dynamics (see [19], [3]). More elaborated parametric Option Pricing Models

(OPMs) that allow for stochastic volatility and jumps have been introduced in an attempt to eliminate some of the BS biases (i.e. [3]). Although these models seem to produce more accurate pricing results compared to the BS, yet, they are quite challenging and complex when used in real time applications. For this reason, the BS is considered to be a significant benchmark model for both academic and practical purposes.

Nowadays, there is a great quest for nonparametric methods and techniques that can potentially alleviate the limitations of parametric OPMs. Practitioners have always a need for more accurate OPMs that can be utilized in real-world applications. Nonparametric methods, such as Artificial Neural Network, Radial Basis Functions, Kernel Regression and other approaches, have been extensively investigated in empirical option pricing applications (see [14], [1], [6], [18], [2] and referenced therein). Support Vector Regression (SVR) is another powerful, nonparametric-data driven, method that is suitable for use in the empirical option pricing area as well. Support Vector Machines have found significant applications in electrical engineering, bioinformatics, pattern recognition, text analysis, computer vision etc (see [20], and references therein). Despite this, they have not gained, yet, any significant popularity in financial econometric applications, with only few studies being the exception to this. For instance, [17] apply them to approximate the noisy Mackey-Glass system and the Santa Fe Times Series Competition; [12] apply such methods for one-step ahead prediction of the weekly 90-day T-bill rate and the daily DAX30 closing prices; and [7] apply SVR to forecast the five day relative difference in percentage of price for five futures contracts.

In this paper, we develop SVR models for pricing European options and compare them with parametric OPMs. We consider the traditional SVR approach as originally developed by Vapnik based on the ε -insensitive loss function (ε -SVR thereafter, see [23]), which is considered to be more robust when noise is non-Gaussian. In addition, we consider the Least Squares Support Vector Regression (LS-SVR), which is a subsequent variant of the original methodology, proposed by Suykens and co-workers (see [21]). LS-SVR can be more robust when noise is Gaussian and it relies on fewer tuning hyper-parameters that can expedite the estimation process. It also minimizes a least squares loss function which is most common in empirical options pricing studies (see [9]).

In this study, we estimate SVR models using two different target functions (desired outputs). One that approximates the unknown empirical option pricing function explicitly, by modeling the market prices of the call options (called the market target function), and one implicitly, by modeling the residual between the actual call market price and the parametric option price estimate (called the hybrid target function). These target functions have been also considered previously in the empirical option pricing research (see [2] and references therein). The SVR models are compared with the parametric BS model using overall average implied parameters and contract specific implied volatility versions derived by the Deterministic Volatility Functions (DVF) approach proposed by [10]. To the

best of our knowledge, this is the first time that such a comprehensive application is considered¹ in the empirical option pricing field.

In the following, we first review the parametric models and the market and hybrid ε -SVR and LS-SVR models. Then we discuss the dataset and the methodologies employed to get the implied parameter estimates. Subsequently we review the numerical results and we conclude.

2 The Parametric Models Used

The BS formula for European call options modified for dividend-paying underlying asset is:

$$c^{BS} = Se^{-d_y T} N(d) - X e^{-rT} N\left(d - \sigma\sqrt{T}\right) \quad (1)$$

$$d = \frac{\ln(S/X) + (r - d_y)T + (\sigma\sqrt{T})^2/2}{\sigma\sqrt{T}} \quad (2)$$

where c^{BS} is premium paid for the European call option, S is the spot price of the underlying asset, X is the exercise price of the call option, r is the continuously compounded risk free interest rate, d_y is the continuous dividend yield paid by the underlying asset, T is the time left until the option expiration date, σ^2 is the yearly variance rate of return for the underlying asset and $N(\cdot)$ stands for the standard normal cumulative distribution.

In this study, we also employ the DVF approach which was proposed by [10]. DVF can be used to estimate per contract volatility for the BS model and it is a practical approach to mitigate the volatility smile anomaly. We estimate the following DVF specification:

$$DVF : \sigma_{DVF}^{BS} = \max(0.01, \alpha_0 + \alpha_1 X + \alpha_2 X^2 + \alpha_3 T + \alpha_4 XT) \quad (3)$$

Based on [10] the above model specification seems to work well for the market under consideration.

3 The Nonparametric Approaches

3.1 ε -Insensitive Support Vector Regression

The idea behind the SVR is to estimate the coefficient values w and b that optimize the generalization ability of the regressor by minimizing the following regularized loss function:

¹ Trafalis et al. (2003) [22] create artificial option pricing data via the Monte-Carlo simulation technique. In order to compare the Black and Scholes equations with the ε -SVR models, they use 1,500 option observations in their out-of-sample tests. In contrast, one of the major contributions of our study is that, we employ market data for the S&P 500 Index options, and we include 21,644 observations in our out-of-sample pricing performance comparisons.

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{j=1}^P L_\varepsilon(t_j, f(x_j)) \quad (4)$$

where t denotes the target function observed in market data and P denotes the number of datapoints considered. In addition, $f(x)$ is the form of the SVR function approximation and is given by:

$$f(x) = w^T \phi(x) + b \quad (5)$$

and $L_\varepsilon(t_j, f(x_j))$ is the so-called Vapnik's ε -insensitive loss functions defined as:

$$L_\varepsilon(t, f(x)) = |t - f(x)|_\varepsilon = \begin{cases} 0 & \text{if } |t - f(x)| \leq \varepsilon \\ |t - f(x_j)| & \text{otherwise} \end{cases} \quad (6)$$

In the above formulations $\phi(x) : R^N \rightarrow R^{N_h}$ represents a nonlinear mapping (transformation) of the input space to an arbitrarily high-dimensional feature space, which can be infinite dimensional (in such case the weights vector will also become infinite dimensional). The constant $C > 0$ determines the trade-off between the amount up to which deviations larger than ε are tolerated and the flatness (complexity) of the estimated model. The estimation of the w and b is done by formulating the following optimization problem in the primal weight space of the unknown coefficients:

$$\min_{w,b,\xi,\psi} L_p(w, \xi, \psi) = \frac{1}{2} w^T w + C \sum_{j=1}^P (\xi_j + \psi_j) \quad (7)$$

subject to

$$\begin{aligned} t_j - w^T \phi(x_j) - b &\leq \varepsilon + \xi_j, \quad j = 1, \dots, P \\ w^T \phi(x_j) - t_j + b &\leq \varepsilon + \psi_j, \quad j = 1, \dots, P \\ \xi_j, \psi_j &\geq 0, \quad j = 1, \dots, P \end{aligned} \quad (8)$$

where ξ_j and ψ_j are defined in the prime space and they are introduced in order to make the solution of the optimization problem feasible for all datapoints that are outside the ε -tube. Transforming the above into its dual formulation² and after applying the kernel trick results to a quadratic programming problem (see [23]). To successfully apply the methodology for nonlinear regression problems it is necessary to apply the kernel trick by choosing a proper kernel function:

$$K(x_j, x_i) = \phi(x_j)^T \phi(x_i) \quad (9)$$

A function that is symmetric, continuous and satisfies Mercer's condition (see [23] for details) is admissible for this case. The Gaussian kernel is a widespread kernel function that is admissible for use with SVR:

$$K(x_j, x_i) = \exp\left(-\frac{\|x_j - x_i\|^2}{2v_K^2}\right) \quad (10)$$

² In nonlinear regression problems the primal weights vector w can become infinite dimensional due to the applied transformation $\phi(x_j)$. For this reason the solution of the problem is better derived via its dual formulation.

where $\|x_j - x_i\|^2$ measures the distance between two datapoints and v_K^2 is called the kernel width parameter and is used as a normalizing factor. It can be shown that when the Gaussian kernel function is considered, the nonlinear mapping $\phi(x_j)$ is infinite dimensional and also that SVR are universal approximators (see [23] for details), an implication of paramount importance that also contributes to the growing popularity of the SVR approach.

3.2 Least Squares Support Vector Machines

The Least Squares Support Vector Machines method is a variant of the ε -SVR originally proposed and developed by Suykens and co-workers (see [21]). According to this approach, the model estimated is given by the following optimization problem in the primal weight space³:

$$\min_{w,b,e} = L_P(w, e) = \frac{1}{2} w^T w + \omega \frac{1}{2} \sum_{j=1}^P e_j^2 \quad (11)$$

subject to

$$t_j = w^T \phi(x_j) + b + e_j, \quad j = 1, \dots, P \quad (12)$$

The above formulation is nothing else but a ridge regression cost function formulated in the featured space defined by the mapping $\phi(x)$. Parameter ω determines again the trade-off between the model complexity and goodness of fit to the estimation data. As in the case of ε -SVR (see [21]), after applying the kernel trick we obtain the following linear system in a^* and b^* :

$$\sum_{j=1}^P (a_j^* K(x_j, x)) + b^* + \frac{a_j^*}{\omega} = t_j, \quad j = 1, \dots, P \quad (13)$$

$$\sum_{j=1}^P a_j^* = 0 \quad (14)$$

where the resulting LS-SVR model that characterizes the estimated regression function is given by:

$$f(x) = \sum_{j=1}^P a_j^* K(x, x_j) + b^* \quad (15)$$

The error variable e_j is used to control deviations from the regression function instead of the slack variables ξ_j , ψ_j and a squared loss function is used instead of the ε -insensitive loss function. This has two implications regarding the solution of the problem: *i*) lack of sparseness since every data point will now be a support

³ We will continue using the notation w and b but the reader should be careful not to confuse these free parameters with the ones used in ε -SVR; although their meaning is closely related the estimation techniques are different.

vector, something that can be considered as a drawback compared to the ε -SVR, and *ii)* only two parameters ω and v_K^2 are needed to be tuned compared to three for ε -SVR; this is an advantage, since it reduces the possible parameters combinations (2-D grid instead of 3-D) and at the same time reduces the risk of selecting a suboptimal parameter combination. Due to the reasons explained above, estimating a set of LS-SVR models can be potentially faster compared to ε -SVR.

4 Data and Methodology

4.1 Data and Filtering Rules

Our dataset covers the period February 2003 to August 2004. The S&P 500 index call options are used because this option market is extremely liquid. They are the most popular index options traded in the CBOE and the closest to the theoretical setting of the parametric models (see [10]). In our analysis, we use the midpoint of the call option bid-ask spread since as noted by [10], using bid-ask midpoints rather than trade prices reduces noise in the cross sectional estimation of implied parameters. Each day, the midpoint of the call option bid ask spread at the close of the market, c^{mrk} , is matched with the closing value of S&P 500 index⁴. To create an informative dataset we employ various filtering rules previously adopted by mainstream papers in this field like the one of [3] (see [2] for further details).

4.2 SVR Hyper-parameters and Data Splitting

Model capacity for SVR models is part of the optimization problem but cross-validation may be needed to properly select the tuning hyper-parameters and to ensure high out-of-sample accuracy. For ε -SVR and LS-SVR, we have conducted a pilot study using data from 2002⁵ in order to determine areas of the tuning parameters values that result to models which performed well out-of-sample. For ε -SVR we examine 40 possible combinations per (weekly) training sample by looking into parameter values in the following areas: $10 \leq C \leq 200$, $0.025 \leq \varepsilon \leq 0.05$ and $1.00 \leq v_K \leq 10.00$. For LS-SVR we examine 30 possible combinations per (weekly) training sample by looking into parameter values in the following areas: $10 \leq \omega \leq 1000$ and $10 \leq v_K \leq 50$.

Regarding the data splitting, our estimating (training) sample is always by using one month of data (around 23 trading days) and our validation sample is always five trading days (one week). After estimating all possible model combinations using the hyper-parameter values, the regression model with the least

⁴ Data synchronicity should be minimal issue for this highly active market (see also [11]). Among others, [9] and [8] use daily closing prices of European call options written on the S&P 500 index.

⁵ Note that 2002 data is not included in our out-of-sample testing. Specifically, our out-of-sample period starts in March 2003.

Root Mean Squared Error (RMSE) in the validation dataset is chosen and used for out-of-sample pricing during the next five trading days (one week). In this paper, the period March 2003 to August 2004 is a period where we can get out-of-sample pricing estimates from all models. For this out-of-sample period we have 21644 datapoints. The focus of our analysis will be based on the RMSE measure since Bates (2000, [4]) points out that it is a relatively intuitive error measure and is useful for comparison with other work in empirical option pricing.

4.3 Implied Parameters for the Black and Scholes Model

The methodology employed here for the estimation of the overall average implied volatility (single per day) is similar to that in previous studies ([3]) that adopt the Whaley's (1982) [24] simultaneous equation procedure to minimize a price deviation function with respect to the unobserved parameters. The above methodology is applied daily to estimate a single overall average implied volatility (σ_{av}^{BS}) and also to estimate the coefficient values of the DVF specification shown in Eq. (3), so as to have a daily unique per contract volatility estimate (σ_{DVF}^{BS}).

4.4 The Set of Alternative Models

With the BS models we use as input S , X , T^6 , d_y^7 , r^8 , and any of the following two volatility estimates: σ_j^{BS} where $j = \{av, DVF\}$, with BS_j denoting the alternative BS parametric models. The dividend adjusted moneyness ratio ($(Se^{-d_y T})/X$) and time to maturity (T) are always inputs to the SVR models. We examine two different target functions. The market target function, which represents actual market prices of call options, and the hybrid target function, which represents the residual between the actual call market price and the parametric option price estimate. The notation here depends on the additional inputs that are used from the parametric models. Specifically, we estimate and examine: $\varepsilon - SVR_{av}^M$, $\varepsilon - SVR_{DVF}^M$, $\varepsilon - SVR_{DVF}^H$ as well as $LS - SVR_{av}^M$, $LS - SVR_{DVF}^M$, $LS - SVR_{DVF}^H$, where the superscripts "M" and "H" are used to denote models estimated based on *Market* and the *Hybrid* target function respectively, and the subscripts "av" and "DVF" denote the volatility used as an additional input.

⁶ Time to maturity is computed assuming 252 days in a year.

⁷ We have collected a daily dividend yield provided by Thomson Datastream. Jackwerth (2000) [15] also assumes that the future dividends for the S&P 500 index can be approximated by a dividend yield.

⁸ Previous studies have used 90-day T-bill rates as approximation of the interest rate. In this study we use nonlinear cubic spline interpolation for matching each option contract with a continuous interest rate, r , that corresponds to the option's maturity. For this purpose, 1, 3, 6, and 12 months constant maturity T-bills rates (collected from the U.S. Federal Reserve Bank Statistical Releases) were considered.

4.5 Analysis of the Out-of-Sample Results

Table 1 exhibits the out-of-sample performance of the benchmark parametric BS model with two different volatility estimates. As expected, we observe that the DVF based BS model (BS_{DVF}) provides better performance than the corresponding overall average one (BS_{av}).

Table 1. Out-of-sample pricing performance of the parametric models

	BS_{av}	BS_{DVF}
RMSE	3.285	2.008

The out-of-sample results for ε -SVR are shown in the upper panel of Table 2 and for LS-SVR in the lower panel of Table 2. First, we observe the hybrid models to perform considerably better than the models estimated with the market target function. Second, the models estimated with σ_{DVF}^{BS} perform better than the models estimated with σ_{av}^{BS} . The most important observation from this table is that the models $\varepsilon - SVR_{DVF}^H$ and $LS - SVR_{DVF}^H$ outperform the parametric alternatives since they have RMSE substantially lower than 2.00. Specifically, RMSE for $\varepsilon - SVR_{DVF}^H$ is equal to 1.623 and for $LS - SVR_{DVF}^H$ is equal to 1.594; as shown in Table 3, these values are lower than the BS_{DVF} 's RMSE in statistical terms as well.

Table 2. Out-of-sample pricing performance for ε -SVR and LS-SVR

	$\varepsilon - SVR_{av}^M$	$\varepsilon - SVR_{DVF}^M$	$\varepsilon - SVR_{DVF}^H$
RMSE	5.944	2.361	1.623

	$LS - SVR_{av}^M$	$LS - SVR_{DVF}^M$	$LS - SVR_{DVF}^H$
RMSE	4.899	2.107	1.594

We should note that in all cases, the performance of the LS-SVR models is better compared to the ε -SVR models. The above does not necessarily imply that LS-SVR is a superior methodology compared to the ε -SVR. One explanation for their superiority regards the naive hyper-parameter selection process we follow. Second, someone should notice that ε -SVR and LS-SVR employ different functional forms to model the problem under investigation and they use different loss functions to measure performance. If the error in the data is governed by a pure Gaussian noise then we may observe LS-SVR that are optimized based on a sum of squares loss function to perform better; ε -SVR can potentially perform better when noise is non-Gaussian (17). In addition, ε -SVR that use inappropriate large values for ε may introduce systematic bias to the estimation and considerably underfit the relationship (17). Nevertheless, the most important fact we must keep from the above analysis is that both SVR methods outperform the benchmark BS model.

Table 3. t-tests for out-of-sample model performance comparison. Values in the upper (lower) diagonal report the Student's (Johnson's modified [16]) t-value regarding the comparison of means of the squared residuals between models in the vertical heading versus models in the horizontal heading. In general, a positive (negative) t-value larger (smaller) than 1.96 (-1.96) indicates that the model in the vertical (horizontal) heading has a larger MSE than the model in the horizontal (vertical) heading at 5% significance level (for 1% significance level use 2.325 and -2.325 respectively).

	BS_{DVF}	$\epsilon - SVR_{DVF}^H$	$LS - SVR_{DVF}^H$
BS_{DVF}	-	4.812	5.257
$\epsilon - SVR_{DVF}^H$	-6.134	-	0.900
$LS - SVR_{DVF}^H$	-6.077	-2.045	-

5 Conclusions

In this paper, we investigate the option pricing performance of ϵ -insensitive Support Vector Regression and Least Squares Support Vector Regression for call options of the S&P 500 index and, we compare it with the withstanding Black and Scholes model. In our view, the results obtained for the Support Vector Regression models are promising enough for the problem under investigation. We expect that under more sophisticated strategies for calibrating the models' hyper-parameters, both methods can improve their out-of-sample performance further.

References

1. Ait-Sahalia, Y., Lo, W.A.: Nonparametric estimation of state-price densities implicit in financial asset prices. *Journal of Finance* 53, 499–547 (1998)
2. Andreou, P.C., Charalambous, C., Martzoukos, S.H.: Pricing and trading european options by combining artificial neural networks and parametric models with implied parameters. *European Journal of Operational Research* 185, 1415–1433 (2008)
3. Bakshi, G., Cao, C., Chen, Z.: Empirical performance of alternative options pricing models. *Journal of Finance* 52, 2003–2049 (1997)
4. Bates, D.S.: Post-'87 crash fears in the s&p 500 futures option market. *Journal of Econometrics* 94, 181–238 (2000)
5. Black, F., Scholes, M.: The pricing of options and corporate liabilities. *Journal of Political Economy* 81, 637–654 (1973)
6. Schittenkopf, C., Dorffner, G.: Risk-neutral density extraction from option prices: Improved pricing with mixture density networks. *IEEE Transactions on Neural Networks* 12, 716–725 (2001)
7. Cao, L.J., Tay, F.E.H.: Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks* 14, 1506–1518 (2003)
8. Chernov, M., Ghysels, E.: Towards a unified approach to the joint estimation objective and risk neutral measures for the purpose of option valuation. *Journal of Financial Economics* 56, 407–458 (2000)

9. Christoffersen, P., Jacobs, K.: The importance of the loss function in option valuation. *Journal of Financial Economics* 72, 291–318 (2004)
10. Dumas, B., Fleming, J., Whaley, R.: Implied volatility functions: Empirical tests. *Journal of Finance* 53, 2059–2106 (1998)
11. Garcia, R., Gencay, R.: Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics* 94, 93–115 (2000)
12. Gestel, T.V., Suykens, J.A.K., Baestaens, D.E., Lambrechts, A., Lanckriet, G., Vandaele, B., Moor, B.D., Vandewalle, J.: Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks* 12, 809–821 (2001)
13. Hull, J.C.: Option, Futures and Other Derivatives. Pearson Prentice Hall (2008)
14. Hutchinson, J.M., Lo, A.W., Poggio, T.: A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance* 49, 851–889 (1994)
15. Jackwerth, J.C.: Recovering risk aversion from option prices and realized returns. *The Review of Financial Studies* 12, 433–451 (2000)
16. Johnson, N.J.: Modified *t*-test and confidence intervals for asymmetrical populations. *Journal of the American Statistical Association* 73, 536–544 (1978)
17. Müller, K., Smola, A., Ratsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Using Support Vector Machines for Time Series Prediction. In: Advances in Kernel Methods: Support Vector Machines. MIT Press, Cambridge (1999)
18. Lajbcygier, P.: Improving option pricing with the product constrained hybrid neural network. *IEEE Transactions on Neural Networks* 15, 465–476 (2004)
19. Rubinstein, M.: Implied binomial trees. *The Journal of Finance* 49, 771–818 (1994)
20. Smola, A., Scholkopf, B.: A tutorial on support vector regression. Technical report, Royal Holloway College, University of London, UK (1998); NeuroCOLT Technical Report, NC-TR-98-030, Royal Holloway College, University of London, UK
21. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific Publishing, Singapore (2002)
22. Trafalis, T.B., Ince, H., Mishina, T.: Support vector regression in option pricing. In: Proceedings of Conference on Computational Intelligence and Financial Engineering. CIFER 2003, Hong Kong, China (2003)
23. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons, New York (1998)
24. Whaley, R.E.: Valuation of american call options on dividend-paying stocks. *Journal of Financial Economics* 10, 29–58 (1982)

Learning SVMs from Sloppily Labeled Data

Guillaume Stempfel and Liva Ralaivola

Laboratoire d’Informatique Fondamentale de Marseille

Aix-Marseille Université

{guillaume.stempfel,liva.ralaivola}@lif.univ-mrs.fr

Abstract. This paper proposes a modelling of Support Vector Machine (SVM) learning to address the problem of learning with *sloppy labels*. In binary classification, learning with sloppy labels is the situation where a learner is provided with labelled data, where the observed labels of each class are possibly noisy (flipped) version of their true class and where the probability of flipping a label y to $-y$ only depends on y . The noise probability is therefore constant and uniform within each class: learning with *positive and unlabeled data* is for instance a motivating example for this model. In order to learn with sloppy labels, we propose **SLOPPYSVM**, an SVM algorithm that minimizes a tailored nonconvex functional that is shown to be a uniform estimate of the noise-free SVM functional. Several experiments validate the soundness of our approach.

1 Introduction

This paper addresses the problem of learning a Support Vector Machine (SVM) from sloppily labelled data, that is from labelled data where the observed labels are possibly noisy versions of the true labels. We focus on binary classification and the noise process that we consider is the one where the probability of flipping a label to the opposite class depends only on the true label; the flipping probabilities (for the +1 and -1 classes) are therefore constant for each class.

Beyond simple theoretical motivations, being able to learn a large margin classifier is of interest for a few practical situations. If the training data at hand are manually labelled, there are reasons to believe that the labelling process is not perfect and that mislabelling may occur, and uniform mislabelling may be seen as a consequence of the tiresomeness of the manual annotating task. Even more interesting is the connection between learning with sloppy labels and semi-supervised learning where the (few) labelled data available come from only one class; this is a common situation in, e.g., bioinformatics. This connection is formalized in Section 2 where it is also discussed how learning with sloppily labelled data may help learn in the multi-instance framework.

In order to tackle the problem of learning an SVM from sloppy labels, we propose to minimize a new objective functional, that is shown to be a *uniform estimate* of the noise-free SVM objective functional. The minimization of this nonconvex functional is done using a classical quasi-Newton minimization algorithm, where the nonconvex functional is rendered differentiable using the

smoothing trick proposed in [3] for the hinge loss. We also propose a heuristic to automatically estimate the noise levels from the training data.

The paper is organized as follows. Section 2 describes the learning setting and shows how it is related to the problem of semi-supervised learning with positive and unlabeled data only. Section 3 describes the new nonconvex objective functional that we propose to minimize together with our heuristic to automatically estimate the noise levels. Finally, Section 4 reports a number of numerical simulations that support the soundness of our proposed learning method.

2 Formal Setting and Motivations

2.1 Notation

Focusing on binary classification, the target space is $\mathcal{Y} = \{-1, +1\}$ and the input space \mathcal{X} is assumed to be a *Hilbert space* with inner product $\langle \cdot, \cdot \rangle$. The family of classifiers we consider is that of zero-bias hyperplanes on \mathcal{X} defined as $\langle \mathbf{w}, \mathbf{x} \rangle = 0$.¹ The class predicted for \mathbf{x} using \mathbf{w} is given by $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$. Even if we describe our results in the context of linear classifiers, they naturally carry over to the case of kernel classifiers.

Given a fixed (unknown) distribution D on $\mathcal{X} \times \mathcal{Y}$, a noise-free sample $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is made of n data independently and identically distributed according to D . Throughout, we assume: $\exists R > 0 : \mathbb{P}_{\mathbf{x} \sim D}(\langle \mathbf{x}, \mathbf{x} \rangle \leq R^2) = 1$.

Let us introduce $\mathcal{N} = \{\boldsymbol{\eta} : \boldsymbol{\eta} = [\eta^+, \eta^-] \in [0, 1]^2, \eta^+ + \eta^- < 1\}$. The sloppy labelling is defined with respect to a noise vector $\boldsymbol{\eta} = [\eta^+, \eta^-] \in \mathcal{N}$ such that a sample $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is corrupted by independently flipping each label y_i to $-y_i$ with probability η^{y_i} , where $\eta^y = \eta^+$ if $y = +1$ and $\eta^y = \eta^-$ otherwise. This noise process can be modeled using a Rademacher vector $\boldsymbol{\sigma} = [\sigma_1 \cdots \sigma_n]$ of size n , with $\mathbb{P}(\sigma_i = -1) = \eta^{y_i} = 1 - \mathbb{P}(\sigma_i = 1)$, to give the noisy version $\mathcal{S}^\sigma = \{(\mathbf{x}_i, \sigma_i y_i)\}_{i=1}^n$ of \mathcal{S} . In this paper, we will assume that the noise vector $\boldsymbol{\eta} \in \mathcal{N}$ is known (and fixed). We discuss in the conclusion how a reliable estimate of this vector can be carried out from the data.

The margin $\gamma : \mathcal{X} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ is defined by $\gamma(\mathbf{w}, \mathbf{x}, y) = y \langle \mathbf{w}, \mathbf{x} \rangle$. $\ell : \mathbb{R} \rightarrow \mathbb{R}^+$ denotes the hinge loss: $\ell(\gamma) = \max(0, 1 - \gamma)$.

Tackled problem. We address the problem of learning a large margin separating hyperplane from $\mathcal{S}^\sigma = \{(\mathbf{x}_i, \sigma_i y_i)\}_{i=1}^n$, where large margin must be understood with respect to the noise-free sample $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$.

2.2 Main Motivation: Asymmetric Semi-supervised Learning

Our primary motivation to study the possibility of learning a large margin classifier from sloppily labelled data is that of being able to perform semi-supervised

¹ The inclusion of a bias term is straightforward and would not change our message. For sake of clarity and conciseness, we choose to consider zero-bias hyperplanes only.

learning when the labelled data only come from one class, which is called *asymmetric semi-supervised learning* [4]. In this situation, the training data can be written as $\mathcal{S} = \mathcal{S}_{+1} \cup \mathcal{S}_{\text{unl}}$ where $\mathcal{S}_{+1} = \{(\mathbf{x}_i, 1)\}_{i=1}^m$ and $\mathcal{S}_{\text{unl}} = \{\mathbf{x}_i\}_{i=m+1}^n$, and where the data from \mathcal{S}_{+1} are supposed to be actual (i.e. noise free) positive data. The strategy that we envision to tackle this problem is to arbitrarily label all the data from \mathcal{S}_{unl} as negative data, to give $\mathcal{S}_{-1} = \{(\mathbf{x}_i, -1)\}_{i=m+1}^n$, which in turn gives rise to a new training set $\underline{\mathcal{S}} = \mathcal{S}_{+1} \cup \mathcal{S}_{-1}$. Hence, as \mathcal{S}_{unl} possibly contained positive instances, the data of $\underline{\mathcal{S}}$ are sloppily labelled data with $\eta^+ > 0$ (the positive data from \mathcal{S}_{unl} are erroneously labelled as -1) while $\eta^- = 0$ (no negative data is labelled as $+1$). This way of tackling the problem of learning from positive and unlabeled data only was successfully undertaken in [4].

Note that the problem of multi-instance learning can also be cast, to some extent, in a problem of learning with sloppy labels. Indeed, in multi-instance learning, the training set is of the form $\mathcal{S} = \{(\underline{\mathbf{x}}_i, y_i)\}$ where each $\underline{\mathbf{x}}_i$ is a *bag of descriptions*, i.e. $\underline{\mathbf{x}}_i = \{\mathbf{x}_i^1, \dots, \mathbf{x}_i^{n_i}\}$ where $\mathbf{x}_i^j \in \mathcal{X}$. A bag is classified as $+1$ if at least one element of the bag is indeed positive. A strategy to address the multi-instance learning problem is simply to break each multi-instance labelled pair $(\underline{\mathbf{x}}_i, y_i)$ in n_i labelled pairs (\mathbf{x}_i^j, y_i) and to consider the question of learning from $\underline{\mathcal{S}} = \{(\mathbf{x}_i^j, y_i)\}_{i,j=1}^{n,n_i}$. Hence, the negative instances from $\underline{\mathcal{S}}$ are indeed negative, whereas some of the positive instances are in fact negative (recall that it suffices for a bag to contain one positive element for it to be positive). If each bag of instances roughly contains the same ratio of positive instances, then the problem is that of learning from the sloppy training set $\underline{\mathcal{S}}$ with $\eta^+ = 0$ and $\eta^- > 0$.

This paper intends to provide a *first step* to envision tackling asymmetric semi-supervised learning with SVMs.

3 Proposed Approach

3.1 SLOPPYSVM: A Version of CSVM Robust to Sloppy Labels

Recall that the CSVM problem for a sample \mathcal{S} of size n writes as (see, e.g., [6])

$$\min_{\mathbf{w}, b} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \frac{C}{n} \sum_{i=1}^n \ell(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle). \quad (1)$$

where $C \in \mathbb{R}^+$ is a regularization parameter. It is clear that when a sloppily labelled dataset \mathcal{S}^σ is fed to the learning algorithm there is no reason for the \mathbf{w} minimizing (1) to be a valid classifier because of the evaluation of the slack/hinge errors, accounted for by the second term of (1). Therefore, if it were possible to estimate the value of the noise-free slack errors, it would be possible to accurately learn a large margin classifier from the noisy data. In this section, we actually show that such an estimation is possible.

Let $\boldsymbol{\eta} = [\eta^+ \ \eta^-] \in \mathcal{N}$ be fixed and let $K_\eta = \frac{1}{1-\eta^+-\eta^-}$ from now on. Let us introduce the mapping $\hat{\ell} : \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}$, with

$$\hat{\ell}(\gamma, y) = K_\eta [(1 - \eta^{-y}) \ell(\gamma) - \eta^y \ell(-\gamma)]. \quad (2)$$

The following lemma holds.

Lemma 1. $\forall i \in \{1, \dots, n\} : \mathbb{E}_{\sigma_i} \hat{\ell}(\sigma_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle, \sigma_i y_i) = \ell(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle)$. Therefore,

$$\mathbb{E}_{\sigma} \frac{1}{n} \sum_{i=1}^n \hat{\ell}(\sigma_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle, \sigma_i y_i) = \frac{1}{n} \sum_{i=1}^n \ell(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle).$$

Proof. The proof is straightforward. Assume that y_i is fixed (hence is the distribution of σ_i). Let us introduce $\gamma_i = y_i \langle \mathbf{w}, \mathbf{x}_i \rangle$. We have

$$\begin{aligned} \mathbb{E}_{\sigma_i} \hat{\ell}(\sigma_i \gamma_i, \sigma_i y_i) &= (1 - \eta^{y_i}) \hat{\ell}(\gamma_i, y_i) + \eta^{y_i} \hat{\ell}(-\gamma_i, -y_i) \\ &= K_\eta [(1 - \eta^{y_i}) [(1 - \eta^{-y_i}) \ell(\gamma_i) - \eta^{y_i} \ell(-\gamma_i)] + \eta^{y_i} [(1 - \eta^{y_i}) \ell(-\gamma_i) - \eta^{-y_i} \ell(\gamma_i)]] \\ &= K_\eta [(1 - \eta^{y_i})(1 - \eta^{-y_i}) \ell(\gamma_i) - \eta^{y_i} \eta^{-y_i} \ell(\gamma_i)] \\ &= K_\eta (1 - \eta^{y_i} - \eta^{-y_i}) \ell(\gamma_i) = \ell(\gamma_i). \end{aligned}$$

The linearity of the expectation gives the second result of the lemma. \square

This lemma says that, for given parameter \mathbf{w} , we can estimate the noise-free slack/fitting errors from the noisy data. Using $\hat{\ell}$ we therefore propose a new version of CSVM based on noisy data. Given a sloppy dataset $\mathcal{S}^\sigma = \{(\mathbf{x}_i, \sigma_i y_i)\}_{i=1}^n$, this new learning strategy, called SLOPPY SVM, aims at solving

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + \frac{C}{n} \sum_{i=1}^n \hat{\ell}(\sigma_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle, \sigma_i y_i). \quad (3)$$

First observe that the objective function of SLOPPY SVM can actually be computed from the sloppy dataset \mathcal{S}^σ . Additionally, note that (a) if the noise rates are such that $\eta^+ = 0$ and $\eta^- = 0$ then problem (3) boils down to the classical CSVM problem (1), (b) the expectation of the objective function of (3) with respect to the noise process is the objective function of (1). Finally, notice that even though the objective function of (1) is convex, that of (3) is not necessarily convex, because of the nonconvexity of $\hat{\ell}$ (see (2)). Figure 1 illustrates the behaviors of the mappings $\hat{\ell}_y$, for $y = 1$ and $y = -1$, defined as:

$$\hat{\ell}_y : \mathbb{R} \rightarrow \mathbb{R}, \quad \gamma \mapsto \hat{\ell}_y(\gamma) = \hat{\ell}(\gamma, y). \quad (4)$$

3.2 Uniform Closeness of the New Functional to Its Expectation

The question we address now is to quantify how close the objective of (3) is to the objective of the corresponding noise free CSVM problem (for the same value of C and the same instances \mathbf{x}_i), i.e. its expectation. We show using concentration inequalities and properties on the Rademacher complexity of (kernel) linear classifier that uniform bounds (with respect to \mathbf{w}) on the closeness of these functionals can be drawn.

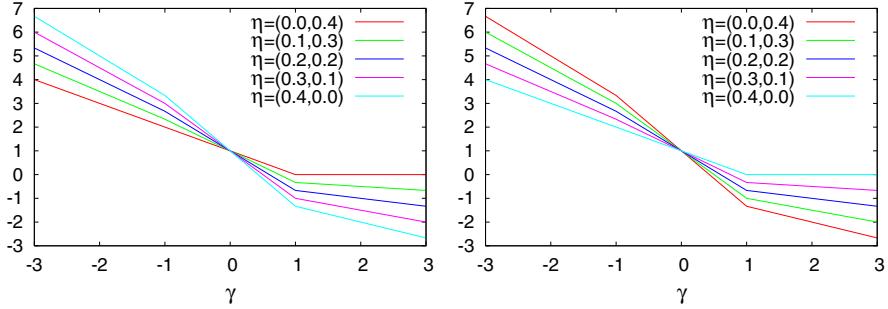


Fig. 1. Left: $\hat{\ell}_{+1}$ (resp. Right: $\hat{\ell}_{-1}$) as a function of γ for different $\boldsymbol{\eta} = [\eta^+ \ \eta^-]$ (see (2)); the $\hat{\ell}_y$'s are nonconvex and Lipschitz with constant $K_\eta(1 + |\eta^+ - \eta^-|)$ (cf. Lemma 2)

To do so, we only consider vectors \mathbf{w} such that $\|\mathbf{w}\| < W$, for some $W > 0$. We are therefore to investigate the closeness of

$$\mu(\mathcal{S}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \ell(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle) \text{ and } \hat{\mu}(\mathcal{S}, \boldsymbol{\sigma}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \hat{\ell}(\sigma_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle, \sigma_i y_i). \quad (5)$$

In order to state our main proposition (Proposition 1), we first introduce and/or recall some results.

Lemma 2. *Let L_η be the constant defined by $L_\eta = 1 - |\eta^+ - \eta^-|$. The mappings $\hat{\ell}_y$, for $y = +1$ and $y = -1$ are Lipschitz with constant $K_\eta L_\eta$.*

Proof. It suffices to observe that the slopes of $\gamma \mapsto \hat{\ell}_y(\gamma)$ are (see also Figure 1), $K_\eta(1 - \eta^{-y})$ if $\gamma < -1$, $K_\eta(1 - \eta^{-y} + \eta^y)$ if $-1 \leq \gamma < 1$, and $K_\eta \eta^y$ otherwise. Observing that $\eta^y \leq 1 - \eta^{-y} + \eta^y$ and $1 - \eta^{-y} < 1 - \eta^{-y} + \eta^y$ together with $1 - \eta^{-y} + \eta^y < 1 + |\eta^+ - \eta^-| = L_\eta$ ends the proof. \square

Theorem 1 (McDiarmid [5]). *Let X_1, \dots, X_n be independent random variables taking values in a set \mathcal{X} , and assume that $f : \mathcal{X}^n \rightarrow \mathbb{R}$ satisfies*

$$\sup_{\substack{\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X} \\ \mathbf{x}'_i \in \mathcal{X}}} |f(\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n) - f(\mathbf{x}_1, \dots, \mathbf{x}'_i, \dots, \mathbf{x}_n)| \leq c_i$$

for every $1 \leq i \leq n$. Then, for every $t > 0$,

$$\mathbb{P}\{|f(X_1, \dots, X_n) - \mathbb{E}f(X_1, \dots, X_n)| \geq t\} \leq 2 \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

Theorem 2 (Bartlett and Mendelson [1]). *Let F be a class of real functions, and let*

$$R_n(F) = \mathbb{E}_{\mathcal{S}, \boldsymbol{\kappa}} \frac{2}{n} \sup_{f \in F} \left| \sum_{i=1}^n \kappa_i f(\mathbf{x}_i) \right|$$

be the Rademacher complexity of F for samples of size n (κ_i are i.i.d random Rademacher variables, i.e. $\mathbb{P}(\kappa_i = +1) = \mathbb{P}(\kappa_i = -1) = \frac{1}{2}$). The following holds:

- If $\varphi : \mathbb{R} \rightarrow \mathbb{R}$ is L_φ -Lipschitz and $\varphi(0) = 0$, then $R_n(\varphi \circ F) \leq 2L_\varphi R_n(F)$
- If $F = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle, \|\mathbf{w}\| \leq W\}$ then $R_n(F) \leq \frac{2WR}{\sqrt{n}}$.

Note that given a distribution D on $\mathcal{X} \times \mathcal{Y}$, the noise process that gives rise to sloppy labels entails a distribution on $\mathcal{X} \times \mathcal{Y} \times \{-1, 1\}$ for triples of the form (X, Y, σ) , with $\mathbb{P}(\sigma = 1 | Y = y) = 1 - \eta^y$; conditioning on $Y\sigma = t$ for $t = \{-1, +1\}$, we may therefore define the conditional distribution $D_{XY\sigma|Y\sigma=t}$ over $\mathcal{X} \times \mathcal{Y} \times \{-1, 1\}$. Then, given vector $\mathbf{t} = [t_1 \dots t_n]$ with $t_i \in \{+1, -1\}$, $D_{S\sigma|\underline{Y\sigma}=\mathbf{t}}^n = \otimes_{i=1}^n D_{XY\sigma|Y\sigma=t_i}$ is the distribution of samples $\mathcal{S} = \{(X_i, Y_i)\}_{i=1}^n$ and noise vector $\sigma = [\sigma_1 \dots \sigma_n]$ such that $Y_i\sigma_i = t_i$. Note that all the triples (X_i, Y_i, σ_i) such that $t_i = 1$ ($t_i = -1$) share the same distribution – and are, of course, independent. The meaning of $D_{S\sigma|\underline{Y\sigma}=\mathbf{t}}^n$ is easily deduced.

Proposition 1. *Let $\mathbf{t} = [t_1 \dots t_n]$ be a fixed vector of $\{-1, +1\}^n$. For all distributions $D, \forall \boldsymbol{\eta} = [\eta^+ \ \eta^-] \in \mathcal{N}, \forall \delta \in (0, 1], \forall \varepsilon \in \mathbb{R}^+$, for all random sample \mathcal{S} and noise vector σ of size n drawn from $D_{S\sigma|\underline{Y\sigma}=\mathbf{t}}^n$ if*

$$n \geq \max \left(\frac{8K_\eta^2 L_\eta^2 (1 + RW)^2}{\varepsilon^2} \ln \frac{4}{\delta}, \frac{256 (2K_\eta L_\eta WR + 1)^2}{\varepsilon^2} \right)$$

then, with probability at least $1 - \delta$,

$$|\mu(\mathcal{S}, \mathbf{w}) - \hat{\mu}(\mathcal{S}, \sigma, \mathbf{w})| < \varepsilon, \quad \forall \mathbf{w} \in \mathcal{X}, \|\mathbf{w}\| \leq W. \quad (6)$$

Proof. Here, expectations must be understood with respect to $D_{S\sigma|\underline{Y\sigma}=\mathbf{t}}^n$ and $D_{S|\underline{Y\sigma}=\mathbf{t}}^n$, n^+ (n^-) is the number t_i equal to 1 (-1). We derive a uniform (wrt $\mathbf{w}, \|\mathbf{w}\| \leq W$) bound on $|\mathbb{E}_{S\sigma} \mu(\mathcal{S}, \mathbf{w}) - \mu(\mathcal{S}, \mathbf{w})|$ and on $|\mathbb{E}_{S\sigma} \hat{\mu}(\mathcal{S}, \sigma, \mathbf{w}) - \hat{\mu}(\mathcal{S}, \sigma, \mathbf{w})|$ (cf equation (5)). It turns out that an adequate sample size for the latter to be lower than $\varepsilon > 0$ is sufficient for the former to be lower than ε as well (the proof omitted for sake of conciseness but it can be easily deduced from the present proof). We thus focus on bounding the function Δ defined as

$$\Delta(\mathcal{S}, \sigma) = \sup_{\mathbf{w} \in \mathcal{X}, \|\mathbf{w}\| \leq W} |\mathbb{E}_{S\sigma} \hat{\mu}(\mathcal{S}, \sigma, \mathbf{w}) - \hat{\mu}(\mathcal{S}, \sigma, \mathbf{w})|.$$

Since $\|\mathbf{w}\| \leq W$ and $\|\mathbf{x}\| \leq R$, the minimum and maximum achievable margin by \mathbf{w} on any pair (\mathbf{x}, y) are $\gamma_{\min} = -RW$ and $\gamma_{\max} = RW$, respectively. Hence, since $\hat{\ell}_y$ is a decreasing function of γ and according to (4), $\hat{\ell}_y$ takes values in the range (both for $y = -1$ and $y = +1$) $[\hat{\ell}_y(\gamma_{\max}); \hat{\ell}_y(\gamma_{\min})]$, which is a subset of

$$[-K_\eta \max(\eta^+, \eta^-)(1 + RW); K_\eta(1 - \min(\eta^+, \eta^-))(1 + RW)].$$

Therefore, using again $\gamma_i = y_i \langle \mathbf{w}, \mathbf{x}_i \rangle$, the maximum variation of $\hat{\ell}_{t_i}(\sigma_i \gamma_i)$ when changing any triple $(\mathbf{x}_i, y_i, \sigma_i)$ to another one is at most

$$K_\eta(1 + RW) [\max(\eta^+, \eta^-) + \max(1 - \eta^+, 1 - \eta^-)] = K_\eta L_\eta (1 + RW).$$

Hence, given the definition of $\hat{\mu}(\mathcal{S}, \boldsymbol{\sigma}, \mathbf{w})$ (cf. (5)), the maximum variation of $\Delta(\mathcal{S}, \boldsymbol{\sigma}, \mathbf{w})$ when changing any $(\mathbf{x}_i, y_i, \sigma_i)$ is at most $K_\eta L_\eta(1 + RW)/n$. Using Theorem II, we thus have (the triples (X_i, Y_i, σ_i) are independent of each other)

$$\mathbb{P}\left\{|\Delta(\mathcal{S}, \boldsymbol{\sigma}) - \mathbb{E}_{\mathcal{S}\boldsymbol{\sigma}}\Delta(\mathcal{S}, \boldsymbol{\sigma})| \geq \frac{\varepsilon}{4}\right\} \leq 2 \exp\left(-\frac{n\varepsilon^2}{8K_\eta^2 L_\eta^2(1 + RW)^2}\right),$$

which is upper bounded by $\delta/2$ for the choice of n stated in the lemma.

Then, we have the following upper bounding on $\mathbb{E}_{\mathcal{S}\boldsymbol{\sigma}}\Delta(\mathcal{S}, \boldsymbol{\sigma})$ (the sup is taken over $\mathbf{w} \in \mathcal{X}, \|\mathbf{w}\| \leq W$), where $\boldsymbol{\kappa}$ is a vector of n independent Rademacher variables and $\mathcal{S}'\boldsymbol{\sigma}' \sim D_{\mathcal{S}\boldsymbol{\sigma}|\underline{\mathbf{Y}\boldsymbol{\sigma}}=\mathbf{t}}^n$, and, thus, $\mathcal{S}' \sim D_{\mathcal{S}|\underline{\mathbf{Y}\boldsymbol{\sigma}}=\mathbf{t}}^n$:

$$\begin{aligned} \mathbb{E}_{\mathcal{S}\boldsymbol{\sigma}}\Delta(\mathcal{S}, \boldsymbol{\sigma}) &= \mathbb{E}_{\mathcal{S}\boldsymbol{\sigma}} \sup |\mathbb{E}_{\mathcal{S}'\boldsymbol{\sigma}'} \hat{\mu}(\mathcal{S}', \boldsymbol{\sigma}', \mathbf{w}) - \hat{\mu}(\mathcal{S}, \boldsymbol{\sigma}, \mathbf{w})| \\ &\leq \mathbb{E}_{\mathcal{S}\boldsymbol{\sigma}} \sup \mathbb{E}_{\mathcal{S}'\boldsymbol{\sigma}'} |s\hat{\mu}(\mathcal{S}', \boldsymbol{\sigma}', \mathbf{w}) - \hat{\mu}(\mathcal{S}, \boldsymbol{\sigma}, \mathbf{w})| \quad (\text{triangle ineq.}) \\ &\leq \mathbb{E}_{\mathcal{S}\boldsymbol{\sigma}\mathcal{S}'\boldsymbol{\sigma}'} \sup |\hat{\mu}(\mathcal{S}', \boldsymbol{\sigma}', \mathbf{w}) - \hat{\mu}(\mathcal{S}, \boldsymbol{\sigma}, \mathbf{w})| \quad (\text{Jensen ineq.}) \\ &= \frac{1}{n} \mathbb{E}_{\mathcal{S}\mathcal{S}'\boldsymbol{\kappa}} \sup \left| \sum_{i=1}^n \hat{\ell}_{t_i}(t_i \langle \mathbf{w}, \mathbf{x}'_i \rangle) - \sum_{i=1}^n \hat{\ell}_{t_i}(t_i \langle \mathbf{w}, \mathbf{x}_i \rangle) \right| \quad (\text{(5)} \text{ and } \mathcal{S}, \mathcal{S}' \sim D_{\mathcal{S}|\underline{\mathbf{Y}\boldsymbol{\sigma}}=\mathbf{t}}^n) \\ &= \frac{1}{n} \mathbb{E}_{\mathcal{S}\mathcal{S}'\boldsymbol{\kappa}} \sup \left| \sum_{i=1}^n \kappa_i (\hat{\ell}_{t_i}(t_i \langle \mathbf{w}, \mathbf{x}'_i \rangle) - \hat{\ell}_{t_i}(t_i \langle \mathbf{w}, \mathbf{x}_i \rangle)) \right| \quad (\mathbf{x}_i \text{ and } \mathbf{x}'_i \text{ are i.i.d.}) \\ &\leq \frac{2}{n} \mathbb{E}_{\mathcal{S}\boldsymbol{\kappa}} \sup \left| \sum_{i=1}^n \kappa_i \hat{\ell}_{t_i}(t_i \langle \mathbf{w}, \mathbf{x}_i \rangle) \right| \quad (\text{triangle ineq.}) \\ &\leq \frac{2}{n} \left[\mathbb{E}_{\mathcal{S}+\boldsymbol{\kappa}} \sup \left| \sum_{i:t_i=+1} \kappa_i \hat{\ell}_{+1}(\langle \mathbf{w}, \mathbf{x}_i \rangle) \right| + \mathbb{E}_{\mathcal{S}-\boldsymbol{\kappa}} \sup \left| \sum_{i:t_i=-1} \kappa_i \hat{\ell}_{-1}(-\langle \mathbf{w}, \mathbf{x}_i \rangle) \right| \right], \end{aligned}$$

where $\mathcal{S}^+ \sim D_{\mathcal{S}|\underline{\mathbf{Y}\boldsymbol{\sigma}}=[1\dots 1]}^{n^+}$ and $\mathcal{S}^- \sim D_{\mathcal{S}|\underline{\mathbf{Y}\boldsymbol{\sigma}}=[-1\dots -1]}^{n^-}$.

To further upper bound $\mathbb{E}_{\mathcal{S}\boldsymbol{\sigma}}\Delta(\mathcal{S}, \boldsymbol{\sigma})$, we introduce $Q_{n^+}(W)$ defined as

$$Q_{n^+}(W) = \frac{2}{n^+} \mathbb{E}_{\mathcal{S}+\boldsymbol{\kappa}} \sup_{\mathbf{w}: \|\mathbf{w}\| \leq W} \left| \sum_{i=1}^{n^+} \kappa_i \hat{\ell}_{+1}(\langle \mathbf{w}, \mathbf{x}_i \rangle) \right|,$$

and we observe that

$$\begin{aligned} Q_{n^+}(W) &\leq \frac{2}{n^+} \mathbb{E}_{\mathcal{S}+\boldsymbol{\kappa}} \sup \left| \sum_{i=1}^{n^+} \kappa_i \hat{\ell}_{+1}(\langle \mathbf{w}, \mathbf{x}_i \rangle) - 1 \right| + \frac{2}{n^+} \mathbb{E}_{\boldsymbol{\kappa}} \left| \sum_{i=1}^{n^+} \kappa_i \right| \quad (\text{triangle ineq.}) \\ &\leq \frac{2}{n^+} \mathbb{E}_{\mathcal{S}+\boldsymbol{\kappa}} \sup \left| \sum_{i=1}^{n^+} \kappa_i \hat{\ell}_{+1}(\langle \mathbf{w}, \mathbf{x}_i \rangle) - 1 \right| + \frac{2}{\sqrt{n^+}}, \end{aligned}$$

since $\mathbb{E}_{\boldsymbol{\kappa}} |\sum_{i=1}^n \kappa_i|^2 = \mathbb{E}_{\boldsymbol{\kappa}} \left[\sum_{i,j=1}^n \kappa_i \kappa_j \right] = n$ and for any random variable X , $0 \leq \mathbb{V}(|X|) = \mathbb{E}(|X|^2) - \mathbb{E}^2(|X|)$ and, thus, $\mathbb{E}(|X|) \leq \sqrt{\mathbb{E}(|X|^2)}$.

From Lemma 2, $\hat{\ell}_{+1}(\gamma) - 1$ is Lipschitz with constant $K_\eta L_\eta$ and $\hat{\ell}_{+1}(0) - 1 = 0$. The first term of the last inequality is the Rademacher complexity of the

class of functions defined by the composition of $\hat{\ell}() - 1$ and the set of zero-bias hyperplanes \mathbf{w} such that $\|\mathbf{w}\| \leq W$. Using Theorem 2, we therefore have:

$$Q_{n^+}(W) \leq \frac{4K_\eta L_\eta WR}{\sqrt{n^+}} + \frac{2}{\sqrt{n^+}} = \frac{2(2K_\eta L_\eta WR + 1)}{\sqrt{n^+}},$$

while a similar inequality holds for the counterpart $Q_{n^-}(W)$ of $Q_{n^+}(W)$. Hence,

$$\mathbb{E}_{S\sigma} \Delta(S, \sigma) \leq \frac{n^+}{n} \frac{2(2K_\eta L_\eta WR + 1)}{\sqrt{n^+}} + \frac{n^-}{n} \frac{2(2K_\eta L_\eta WR + 1)}{\sqrt{n^-}} \leq \frac{4(2K_\eta L_\eta WR + 1)}{\sqrt{n}}$$

which, for the value of n stated in the lemma is upper bounded by $\varepsilon/4$. Therefore, with probability at least $1 - \delta/2$ the following holds uniformly over \mathbf{w} , $\|\mathbf{w}\| \leq W$:

$$|\mathbb{E}_{S\sigma} \hat{\mu}(S, \sigma, \mathbf{w}) - \hat{\mu}(S, \sigma, \mathbf{w})| \leq \frac{\varepsilon}{4} + \mathbb{E}_{S\sigma} \Delta(S, \sigma) \leq \frac{\varepsilon}{4} + \frac{\varepsilon}{4} = \frac{\varepsilon}{2}.$$

Likewise, $|\mathbb{E}_{S\sigma} \mu(S, \mathbf{w}) - \mu(S, \mathbf{w})| \leq \varepsilon/2$ with probability $1 - \delta/2$ as well. Noting that $\mathbb{E}_{S\sigma} \mu(S, \mathbf{w}) = \mathbb{E}_S \mu(S, \mathbf{w}) = \mathbb{E}_{S\sigma} \hat{\mu}(S, \sigma, \mathbf{w})$ and using

$$|\mu(S, \mathbf{w}) - \hat{\mu}(S, \sigma, \mathbf{w})| \leq |\mathbb{E}_{S\sigma} \mu(S, \mathbf{w}) - \mu(S, \mathbf{w})| + |\mathbb{E}_{S\sigma} \hat{\mu}(S, \sigma, \mathbf{w}) - \hat{\mu}(S, \sigma, \mathbf{w})|$$

ends the proof. \square

Proposition 2. *With the same hypotheses as in Proposition 1, for S drawn according to D (no conditioning on $\mathbf{Y}\sigma$), and corresponding random vector σ , the following holds with probability at least $1 - \delta$:*

$$|\mu(S, \mathbf{w}) - \hat{\mu}(S, \sigma, \mathbf{w})| < \varepsilon, \quad \forall \mathbf{w} \in \mathcal{X}, \|\mathbf{w}\| \leq W. \quad (7)$$

Proof. It Φ denotes the event given by equation (6), then we just stated that $\forall \mathbf{t} \in \{-1, +1\}^n$, $\mathbb{P}_{S\sigma \sim D_{S\sigma|\underline{Y}\sigma=\mathbf{t}}^n}(\Phi) \geq 1 - \delta$. Then,

$$\begin{aligned} \mathbb{P}_{S \sim D}(\Phi) &= \mathbb{E}_{\mathbf{T}} [\mathbb{E}_{S\sigma \sim D_{S\sigma|\underline{Y}\sigma=\mathbf{T}}^n} \mathbb{I}_{\Phi}] = \sum_{\mathbf{t}} \mathbb{E}_{S\sigma \sim D_{S\sigma|\underline{Y}\sigma=\mathbf{t}}^n} \mathbb{I}_{\Phi} \mathbb{P}(\mathbf{Y}\sigma = \mathbf{t}) \\ &= \sum_{\mathbf{t}} \mathbb{P}_{S\sigma \sim D_{S\sigma|\underline{Y}\sigma=\mathbf{t}}^n}(\Phi) \mathbb{P}(\mathbf{Y}\sigma = \mathbf{t}) \geq \sum_{\mathbf{t}} (1 - \delta) \mathbb{P}(\mathbf{Y}\sigma = \mathbf{t}) = 1 - \delta. \quad \square \end{aligned}$$

This establishes the closeness of the functional of SLOPPYSVM to the original SVM functional and justifies the strategy that consists in minimizing (3), when sloppily labeled data are available.

3.3 A Quasi-Newton Optimization Method

In order to actually minimize (3), we use the BFGS quasi-Newton minimization procedure directly applied to the primal. As proposed by [3], we used a twice-differentiable approximation L_h of the hinge loss function ℓ , with, for $h > 0$

$$L_h(\gamma) := \begin{cases} 0 & \text{if } \gamma > 1 + h \\ \frac{(1+h-\gamma)^2}{4h} & \text{if } |1 - \gamma| \leq h \\ 1 - \gamma & \text{if } \gamma < 1 - h \end{cases} \quad (8)$$

Plugging in this loss function in (3) as a surrogate for ℓ , we end up with an unconstrained minimization problem for which it is easy to find a local minimum.

Table 1. Generalization errors (and standard deviations) obtained from simulations conducted on 5 datasets from the UCI repository when corrupted by various amount of noise. Italicized results indicate best errors (not necessary statistically significant).

$[\eta^+, \eta^-]$	Alg.	<i>Diabetis</i>	<i>Flare-Solar</i>	<i>German</i>	<i>Heart</i>	<i>Splice</i>
[0, 0]	All	23.3 ± 1.59	32.32 ± 1.81	24.03 ± 2.24	15.31 ± 3.17	16.11 ± 0.56
[0, 0.2]	CSVM	25.18 ± 2.08	38.99 ± 6.15	25.32 ± 2.65	18.11 ± 4.02	19.20 ± 1.02
	SLOPPY	23.69 ± 1.74	32.45 ± 1.73	24.67 ± 2.04	16.19 ± 3.78	17.10 ± 0.70
[0, 0.4]	CSVM	44.06 ± 11.9	44.74 ± 1.81	44.77 ± 7.81	30.10 ± 6.48	29.64 ± 2.14
	SLOPPY	24.02 ± 1.79	33.91 ± 4.08	26.06 ± 2.47	17.68 ± 3.88	18.71 ± 0.68
[0.2, 0]	CSVM	25.00 ± 2.37	37.45 ± 3.02	27.93 ± 3.12	18.78 ± 3.9	18.24 ± 1.10
	SLOPPY	23.63 ± 1.64	32.78 ± 1.66	24.40 ± 2.31	16.63 ± 4.01	16.93 ± 0.10
[0.2, 0.4]	CSVM	33.83 ± 9.93	44.57 ± 2.52	37.39 ± 6.39	29.97 ± 8.10	28.08 ± 2.17
	SLOPPY	26.04 ± 2.96	35.65 ± 4.32	28.32 ± 3.14	20.73 ± 5.27	23.67 ± 1.73
[0.4, 0]	CSVM	33.93 ± 2.93	41.43 ± 2.89	29.82 ± 1.95	27.97 ± 6.42	28.3 ± 2.14
	SLOPPY	24.1 ± 1.91	33.37 ± 2.71	25.36 ± 2.4	17.18 ± 3.75	18.08 ± 0.90
[0.4, 0.2]	CSVM	33.49 ± 3.56	40.61 ± 3.5	29.82 ± 1.95	26.96 ± 7.06	27.84 ± 2.42
	SLOPPY	26.29 ± 2.80	35.85 ± 4.15	28.80 ± 2.96	21.66 ± 7.08	23.33 ± 2.08

4 Numerical Simulations

In order to illustrate the behavior of SLOPPYSVM on real data, we have carried out simulations on 5 UCI datasets, preprocessed and made available by Gunnar Rätsch². These numerical simulations consist in adding a class dependent noise to the data, and then learn on the sloppy datasets with CSVM, and SLOPPYSVM, the actual noise levels being provided to SLOPPYSVM. For each problem, the data are split into 100 training and test sets (except for Splice, where only 20 replicates are available). All the experiments have been made using a linear kernel, as they make it possible in the noise free case to achieve nearly 'optimal' accuracies. Several values (from 0.005 to 1000) of the parameter C have been tested. For each problem and for each noise level, we picked the value of C that minimizes the generalization error of CSVM. For SLOPPYSVM, whichever the level of noise, we use the value of C that minimizes the generalization error when CSVM is trained on the noise-free data.

The results of the simulations are reported in Table II, which shows the error rates and the standard deviations computed on the replicates of the different problems. It is clear that, even if the behavior of CSVM may seem satisfactory on some problems, especially when η^+ and η^- are close to each other, CSVM is consistently outperformed by SLOPPYSVM on all the conducted experiments. The performances of SLOPPYSVM are close, even under rather high noise levels, to the performance obtained by CSVM on noise-free datasets, while keeping a relatively low standard deviation. This shows the stability of SLOPPYSVM.

² <http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm>

5 Conclusion

We have proposed SLOPPYSVM, a new large margin algorithm that can deal with sloppily labeled data. It makes use of a new objective function that computes an estimator the noise-free fitting term of usual CSVM. A theoretical analysis ensures the closeness of the proposed objective functional to the noise-free one. Using a standard quasi-Newton minimization procedure, numerical simulations conducted on noisy problems prove the soundness of our approach.

We have carried out preliminary experiments on asymmetric semi-supervised learning problems where the noise is estimated from the data. The procedure of noise estimation is that proposed in [2] for co-training, which is proved to be consistent: among candidate noise vectors, this suggests to select the noise η that minimizes the quantity $\mathbb{P}(h(x) = 1|y\sigma = -1) + \mathbb{P}(h(x) = -1|y\sigma = 1)$, where h is a learned SLOPPYSVM using η . The results we obtain are very promising and we plan to investigate this empirical evaluation more thoroughly.

References

1. Bartlett, P.L., Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. *J. of Machine Learning Research* 3, 463–482 (2002)
2. Blum, A., Mitchell, T.: Combining Labeled and Unlabeled Data with Co-Training. In: Proc. of the 11th Conf. on Computational Learning Theory, pp. 92–100 (1998)
3. Chapelle, O.: Training a support vector machine in the primal. *Neural Comput.* 19(5), 1155–1178 (2007)
4. Magnan, C.: Asymmetrical Semi-Supervised Learning and Prediction of Disulfide Connectivity in Proteins. *RIA, New Methods in Machine Learning: Theory and applications* 20(6), 673–695 (2006)
5. McDiarmid, C.: On the method of bounded differences. In: *Survey in Combinatorics*, pp. 148–188 (1989)
6. Schölkopf, B., Smola, A.J.: *Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond*. MIT University Press, Cambridge (2002)

The GMM-SVM Supervector Approach for the Recognition of the Emotional Status from Speech

Friedhelm Schwenker¹, Stefan Scherer¹, Yasmine M. Magdi²,
and Günther Palm¹

¹ University of Ulm

Institute of Neural Information Processing
89069 Ulm, Germany

{friedhelm.schwenker,stefan.scherer,guenther.palm}@uni-ulm.de

² German University in Cairo

Computer Science and Engineering Department
38 Beirut st. - Heliopolis, 11341 Cairo, Egypt

Abstract. Emotion recognition from speech is an important field of research in human-machine-interfaces, and has various applications, for instance for call centers. In the proposed classifier system RASTA-PLP features (perceptual linear prediction) are extracted from the speech signals. The first step is to compute an universal background model (UBM) representing a general structure of the underlying feature space of speech signals. This UBM is modeled as a Gaussian mixture model (GMM). After computing the UBM the sequence of feature vectors extracted from the utterance is used to re-train the UBM. From this GMM the mean vectors are extracted and concatenated to the so-called GMM supervectors which are then applied to a support vector machine classifier. The overall system has been evaluated by using utterances from the public Berlin emotional database. Utilizing the proposed features a recognition rate of 79% (utterance based) has been achieved which is close to the performance of humans on this database.

1 Introduction

Research in affective computing aim to provide simpler and more natural interfaces for human-computer interaction applications. Detecting and recognizing the emotional status of a user is important in designing and developing efficient and productive human-computer interaction interfaces [5]. Emotion analysis and processing is a multi-disciplinary topic, which has been emerging as a rich research area in recent times [7][17][21]. Speech can be used for emotion recognition which is not only simple to process, but can also be incorporated into the existing speech processing applications [5][9][20].

One of the main issues in designing an automatic emotion recognition system is the selection of the features that can represent the corresponding emotions. In [16], pitch and linear predictive coding (LPC) features were used as input

to an artificial neural network (ANN). After detecting the on- and off-sets of the utterances, a 300 dimensional vector was used, which resulted in classification rates of around 50% detecting eight different emotions. In earlier work multi classifier systems (MCS) were trained with three feature types, comprising modulation spectrum, relative spectral transform - perceptual linear prediction (RASTA-PLP), and perceived loudness features, to recognize seven different emotions with an accuracy of more than 70% [21]. The Mel Frequency Cepstral Coefficients (MFCC) based features were used in [15], which were obtained with a window of 25 ms sampled every 10 ms. The Hidden Markov Model (HMM) was then used for training each of the four targeted emotions. After training a combination of all the phoneme class based HMMs on the TIMIT database, for each of the emotions, the classification performance reached around 76%. In [6], k-nearest neighbor (KNN) algorithm was applied to classify four emotions, which resulted in 65 % accuracy. The pitch based statistics, such as contour of pitch, maximum, minimum, and slope were considered as features. Broadly speaking, differences in the various approaches arise from the overall goal (recognizing single vs. multiple emotions), the specific features used, and the classification framework. The anger vs. neutral emotion classification was studied, particularly in the context of interactive voice response systems with specific application to call centers in [23]. 37 prosody features related to pitch, energy, and duration were used as features, and for classification neural networks, SVM, and KNN were applied. With the most significant feature set of 19 features, the best recognition accuracy about 90% was achieved using SVMs.

The Gaussian Mixture Models (GMM) supervector approach has been used for text-independent speaker recognition tasks. The key idea of this approach is to re-train a universal background model realized by a GMM by utilizing the current utterance through maximum a-posteriori (MAP) adaptation, and to use the adapted means of the mixture components as supervectors for speaker recognition [4]. In this paper we propose RASTA-PLP features which might be computed by standard signal processing techniques. These features are biologically inspired since perceptually scaled filtering is used, and they have been applied to various speech processing tasks [12]. In Sections 3, 4, and 5 GMMs, the GMM supervector approach, and support vector machines (SVM) are introduced. Finally, the recognition system is discussed and the results are shown in Section 7.

2 Feature Extraction

The feature extraction proposed in this paper is composed of tow independent feature streams, namely RASTA-PLP coefficients [12], often used in speech recognition besides MFCC, and spectral energy modulation features [11]. Both are motivated by the human auditory system, which is not equally sensitive to every frequency. Frequencies in the range of around 100 to 2000 Hz are perceived more intensely as frequencies above.

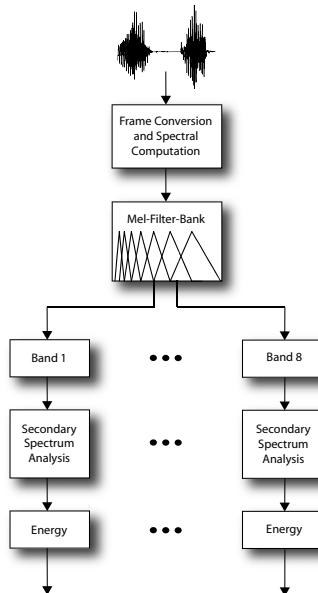


Fig. 1. Schematic description for feature extraction

2.1 RASTA Perceptual Linear Prediction

The PLP feature extraction is based on short term spectrum of speech. PLP is sensitive towards short-term spectral value changes. However, human speech perception does not seem to be that sensitive. For this reason in his work Hermansky proposed the RASTA-PLP approach, which renders the features more robust to linear spectral distortions [2].

2.2 Modulation Spectral Features

Short term analysis of the speech signal, such as extracting spectral features from frames not more than several milliseconds, dominates speech processing for many years. However, these features are strongly influenced by environmental noise and are therefore unstable. In [11], it is suggested to use the so called modulation spectrum of speech to obtain information about the temporal dynamics of the speech signal to extract reliable cues for the linguistic context. Since emotion in speech is often communicated by varying temporal dynamics in the signal the same features are used to classify emotional speech in the following experiments [20].

The proposed features are based on long term modulation spectrum. In this work, the features based on slow temporal evolution of the speech are used to represent the emotional status of the speaker. These slow temporal modulations of speech emulate the perception ability of the human auditory system. Earlier studies reported that the modulation frequency components from the range

between 2 and 16 Hz, with dominant component at around 4 Hz, contain important linguistic information [10,8,14]. Dominant components represent strong rate of change of the vocal tract shape. This particular property, along with the other features has been used to discriminate speech and music [19]. In this work, the proposed features are based on this specific characteristic of speech, to recognize the emotional state of the speaker.

The block diagram for the feature extraction for a system to recognize emotions is shown in Fig. 1. The fast Fourier transform (FFT) for the input signal $x(t)$ is computed over N points with a shift of n samples, which results in a $\frac{N}{2}$ dimensional FFT vector. Then, the Mel-scale transformation, motivated by the human auditory system, is applied to these vectors. The Mel-filter bank with eight triangular filters $H_i[k]$, is defined by:

$$H_i[k] = \begin{cases} \frac{2(k-b_i)}{(d_i-b_i)(c_i-b_i)} & b_i \leq k \leq c_i \\ \frac{2(d_i-k)}{(d_i-b_i)(d_i-c_i)} & c_i \leq k \leq d_i \end{cases}, \quad (1)$$

where $i = 1, \dots, 8$ indicates the index of the i -th filter. b_i and d_i indicate the frequency range of filter H_i and the center frequency c_i is defined as $c_i = (b_i + d_i)/2$. These ranges are equally distributed in the Mel-scale, and the corresponding frequencies b_i and d_i are listed in Table 1. For $k < b_i$ and $k > d_i$ $H_i[k] = 0$.

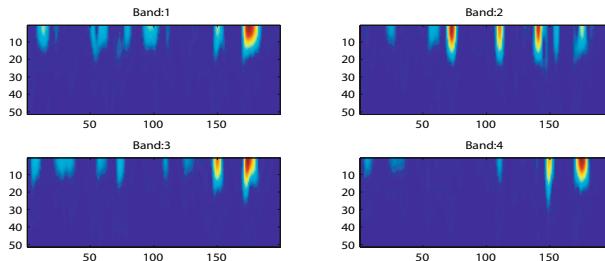


Fig. 2. Modulation spectrum for the first four bands of a single angry utterance. The x-axis represents the time scale, in frames and the y-axis, the frequency in Hz.

For each of the bands, the modulations of the signal are computed by taking FFT over the P points, shifted by p samples, resulting in a sequence of $\frac{P}{2}$ dimensional modulation vectors. Most of the prominent energies can be observed within the frequencies between 2 - 16 Hz. Figure 2 illustrates the modulation spectrum based energies for a single angry utterance, for the values $N = 512$, $n = 160$, $P = 100$ and $p = 1$ for the first four bands. For the classification task following values were used: $N = 1600$, $n = 640$, $P = 10$, $p = 1$. Since the signal is sampled with 16 kHz, N corresponds to 100 ms and n to 40 ms resulting in a feature extraction frequency of 25 Hz. According to the window size P a lead time of 400 ms is necessary. Therefore, one feature vector in the modulation spectrum takes 400 ms into account with an overlap of 360 ms, due to p .

Table 1. Start and end frequencies of the triangular Mel-filters

Band	Start Freq. (Hz)	End Freq. (Hz)
1	32	578
2	257	964
3	578	1501
4	966	2217
5	1501	3180
6	2217	4433
7	3180	6972
8	4433	8256

3 Gaussian Mixture Models

GMM is a semi-parametric estimation technique used to estimate the probability density function (PDF) from a set of data points drawn from this function. The PDF is regarded as a linear combination of M Gaussian functions (components of the model). The value of the PDF at a point \mathbf{x}_i in the d -dimensional space is given by

$$p(\mathbf{x}_i) = \sum_{m=1}^M P(m) \cdot p(\mathbf{x}_i|m) \quad (2)$$

under the constraints:

$$0 \leq P(m) \leq 1 \quad \sum_{m=1}^M P(m) = 1 \quad \int_{\mathbb{R}^d} p(\mathbf{x}|m) d\mathbf{x} = 1 \quad (3)$$

$P(m)$ is the weight (prior probability) of component m and $p(\mathbf{x}_i|m)$ is the value of the Gaussian function described by component m , and can be calculated as follows

$$p(\mathbf{x}_i|m) = \frac{1}{(2\pi)^{d/2}} \cdot |\boldsymbol{\Sigma}_m|^{-1/2} \cdot e^{-\frac{1}{2} \cdot (\mathbf{x}_i - \boldsymbol{\mu}_m)^T \cdot \boldsymbol{\Sigma}_m^{-1} \cdot (\mathbf{x}_i - \boldsymbol{\mu}_m)} \quad (4)$$

The GMM is completely defined by the means of its components, their covariance matrices and their weights. In order to estimate these parameters the well known EM-algorithm is used [1].

4 GMM Supervectors

The feature vectors extracted from speech are passed to classifiers in order to recognize the different emotions. Usually different utterances have different durations leading to feature vectors of different lengths. Since almost all classifiers require constant length vectors as inputs, the stream of feature vectors must be transformed in such an input vector of constant length (see Fig. 3). This is the idea of GMM supervectors which are calculated as follows:

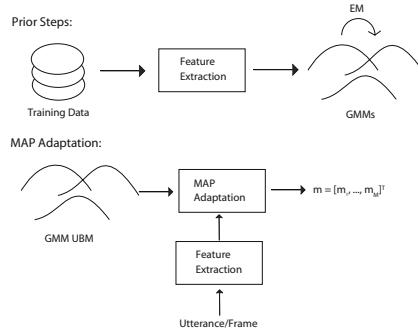


Fig. 3. Calculation of GMM supervectors

1. Collect a general large speech database for the universal background model (UBM). In this study the available data is divided into training and testing sets according to 10-fold cross validation test and the database was built by gathering all the utterances from the current training data set.
2. Calculate the UBM. This UBM is a GMM calculated using the general database. The UBM acts as a basic model that is independent of any emotion, speaker, etc. It is used in order to guarantee a general representation for all possible utterances. Basically, this database should be a large set of speech samples covering different speakers (male or female), in the state of different emotions, in different environments (for instance different types of background noise).
3. For each utterance the pre-trained UBM is adapted in order to represent the information carried by the current utterance. Adaptation is performed through the maximum-a-posteriori (MAP) algorithm.
4. Construct the supervector by concatenating the means of the Gaussian mixture components of the adapted model according:

$$\boldsymbol{\mu} = [\boldsymbol{\mu}_1^T \dots \boldsymbol{\mu}_M^T] \quad (5)$$

these vectors $\boldsymbol{\mu}$ - the GMM supervectors - are then the input vectors to the subsequently following classifiers.

The basic steps of MAP are as follows [2]:

1. Calculate the posterior probability of each component of the model given the current utterance. This corresponds to the probability that this component has contributed to generating the utterance:

$$n_m = \sum_{t=1}^T P(m|\mathbf{x}_t) \quad (6)$$

here m is a Gaussian component and \mathbf{x}_t is an interval of the utterance that currently is used to adapt the UBM, assuming that the utterance is divided into T intervals. The factor $P(m|\mathbf{x}_t)$ is calculated through

$$P(m|\mathbf{x}_t) = \frac{P(m)f(\mathbf{x}_t|m)}{\sum_{k=1}^M P(k)f(\mathbf{x}_t|k)} \quad (7)$$

here $f(\mathbf{x}_t|m)$ is the value of the Gaussian function representing component m at point \mathbf{x}_t .

2. Calculate the new estimate of the mean and covariance parameters, according to the equations

$$\mathbf{E}_m(\mathbf{x}) = \frac{1}{n_m} \sum_{t=1}^T P(m|\mathbf{x}_t) \mathbf{x}_t \quad (8)$$

$$\mathbf{E}_m(\mathbf{x}\mathbf{x}^T) = \frac{1}{n_m} \sum_{t=1}^T P(m|\mathbf{x}_t) \mathbf{x}_t \mathbf{x}_t^T \quad (9)$$

3. Adapt the UBM parameters to represent the features of the current utterance using the equations

$$\hat{P}(m) = [\alpha_m n_m / T + (1 - \alpha_m) P(m)]\gamma \quad (10)$$

$$\hat{\boldsymbol{\mu}}_m = \alpha_m \mathbf{E}_m(\mathbf{x}) + (1 - \alpha_m) \boldsymbol{\mu}_m \quad (11)$$

$$\hat{\boldsymbol{\Sigma}}_m = \alpha_m \mathbf{E}_m(\mathbf{x}^2) + (1 - \alpha_m) (\boldsymbol{\Sigma}_m + \boldsymbol{\mu}_m^2) - \boldsymbol{\mu}_m^2 \quad (12)$$

here $\hat{P}(m)$, $\hat{\boldsymbol{\mu}}_m$ and $\hat{\boldsymbol{\Sigma}}_m$ are the adapted weight, mean and covariance matrix of component m , respectively.

The factor γ in equation (10) ensures that the weights sum up to one, so that the constraints in Eq 3 are satisfied. The adaptation coefficient α_m is a factor balancing the adapted values between old estimates (from the UBM) and calculated parameters (by using MAP adaptation). The adaptation coefficient is calculated using the following equation $\alpha_m = \frac{n_m}{n_m + r}$ here r is the relevance factor.

5 Support Vector Machines

Basically, SVMs are binary classifier that can be extended to multi-class SVMs. Two main decomposition techniques have been used in our study, one-against-one decomposition and one-against-all decomposition. Since the performance of these two decomposition schemes was very similar in this application, the results of the one-versus-rest scheme are presented (see [22] for an overview).

6 Database Description

The Berlin Database of Emotional Speech is used as a test bed for our approach. This corpus is a collection of around 530 utterances spoken in seven different emotions: anger, boredom, disgust, fear, happiness, sadness, and neutral [3]. The

database is publicly available at <http://pascal.kgw.tu-berlin.de/emodb/>. Ten professional actors (five male and five female) read the predefined utterances in an anechoic chamber, under supervised conditions. The text was taken from everyday life situations, and did not include any emotional bias. The utterances are available at a sampling rate of 16 kHz with a 16 bit resolution and mono channel. A human perception test to recognize various emotions with 20 participants resulted in a mean accuracy of around 84% [3].

7 Numerical Evaluation

RASTA-PLP features were extracted from the utterances of the Berlin emotional database. This results in 530 sequences of feature vectors, the length of a sequence depends on the duration of the utterance and ranges from 150 to 500 feature vectors per utterance. The set of utterances has been divided into training and testing sets according to a 10-fold cross validation procedure. For each fold a UBM was constructed using the extracted feature vectors from the fold's training utterances. For each utterance the pre-trained UBM is adapted in order to reflect the individual information and structure of the current utterance, and the mean vectors of the components of an adapted GMM are concatenated to form the supervectors. Nonlinear support vector machines (with the Gaussian kernel function and the one-versus-rest decomposition scheme) are then trained from the collected supervectors of the current fold.

Different numerical tests have been performed on the GMM and the SVM. For the GMM various tests concerning the number of Gaussian components and the type of covariance matrix to be used in the GMM kernel function have been performed. SVM parameters testing included the type of multi-class decomposition scheme, the optimization procedure, the kernel type, the kernel parameter(s) and the hyper-parameter(s) of the SVM approach. As a result of this preprocessing diagonal covariance matrices have been chosen for the GMM components. Diagonal covariance matrices are suitable for many applications since they are a compromise between the full covariance matrix and the spherical model with a single real width parameter. Diagonal models are computationally efficient and usually they lead to more robust GMM estimates than full covariance matrices because smaller training data sets can be used [2]. Based on the numerical test, $m = 40$ Gaussian components have been used for the RASTA features.

The one-against-one approach for the decomposition of the multi-class classification problem, and the sequential minimal optimization (SMO), see [18] for details, to solve the quadratic optimization problem have been used. Furthermore, the radial basis function kernel is used in the system. The kernel parameter has been optimized and was chosen to $\sigma^2 = 3$. Different hyper-parameters C have been tested, and based on these test C was set equal to 10. Using these parameter settings the RASTA-based system scores a mean accuracy of 79% on the basis of 10 runs of 10-cross validation tests. This accuracy scored by a RASTA-based system approaches close to the human recognition of emotions from the same database which is 84% [21].

8 Conclusion

In this work, an emotion recognition system for spoken speech employing the idea of GMM supervectors as a preprocessor of data and SVM as a classifier has been developed (recognition accuracy of 79% utilizing RASTA-PLP features). It was the first time that the GMM supervector approach together with the biological inspired RASTA-PLP features has been used. The data provided by the Berlin database is free of noise. For further research, it is recommended to apply the proposed technique to speech data recorded from a telephone line or in noisy environments. GMM supervectors offer a rather flexible processing scheme for the classification of any type of sequential data, the approach is not limited to speech signals only, for instance it might be applied for image processing tasks such as the recognition of facial expressions.

Acknowledgements

This paper is based on work done within the Transregional Collaborative Research Centre SFB/TRR 62 *Companion-Technology for Cognitive Technical Systems* funded by the German Research Foundation (DFG).

References

1. Bilmes, J.A.: A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, International Computer Science Institute and Computer Science Division, Department of Electrical Engineering and Computer Science, U.C. Berkeley (1998)
2. Bimbot, F., Bonastre, J.-F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-Garcia, J., Petrovska-Delacrètaz, D., Reynolds, D.A.: A Tutorial on Text-Independent Speaker Verification. EURASIP Journal on Applied Signal Processing, 430–451 (2004)
3. Burkhardt, F., Paeschke, A., Rolfs, M., Sendlmeier, W., Weiss, B.: A database of german emotional speech. In: Proceedings of Interspeech 2005 (2005)
4. Campbell, W.M., Sturim, D.E., Reynolds, D.A.: Support vector machines using gmm supervectors for speaker verification support vector machines using gmm supervectors for speaker verification. IEEE Signal Processing Letters 13(5), 308–311 (2006)
5. Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., Taylor, J.G.: Emotion recognition in human-computer interaction. IEEE Signal Processing Magazine 18(1), 32–80 (2001)
6. Dellaert, F., Polzin, T., Waibel, A.: Recognizing emotion in speech. In: Proceedings of ICSLP, pp. 1970–1973 (1996)
7. Devillers, L., Vidrascu, L., Lamel, L.: Challenges in real-life emotion annotation and machine learning based detection. Neural Networks 18, 407–422 (2005)
8. Drullman, R., Festen, J., Plomp, R.: Effect of reducing slow temporal modulations on speech reception. Journal of the Acoustic Society 95, 2670–2680 (1994)
9. Fragapanagos, N., Taylor, J.G.: Emotion recognition in human-computer interaction. Neural Networks 18, 389–405 (2005)

10. Hermansky, H.: Auditory modeling in automatic recognition of speech. In: Proceedings of Keele Workshop (1996)
11. Hermansky, H.: The modulation spectrum in automatic recognition of speech. In: Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (1997)
12. Hermansky, H., Morgan, N.: Rasta processing of speech. IEEE transactions on speech and audio processing 2(4) (October 1994)
13. Kandovan, R.S., Lashkari, M.R.K., Etemad, Y.: Optimization of speaker verification using adapted GaSPPR 2007ussian mixture models for high quality databases. In: Proceedings of the Fourth conference on IASTED International Conference, Anaheim, CA, USA, pp. 264–268. ACTA Press (2007)
14. Kanederaa, N., Araib, T., Hermansky, H., Pavele, M.: On the relative importance of various components of the modulation spectrum for automatic speech recognition. *Speech Communications* 28, 43–55 (1999)
15. Lee, C.M., Yildirim, S., Bulut, M., Kazemzadeh, A., Busso, C., Deng, Z., Lee, S., Narayanan, S.S.: Emotion recognition based on phoneme classes. In: Proceedings of ICSLP 2004 (2004)
16. Nicholson, J., Takahashi, K., Nakatsu, R.: Emotion recognition in speech using neural networks. *Neural Computing and Applications* 9, 290–296 (2000)
17. Oudeyer, P.-Y.: The production and recognition of emotions in speech: features and algorithms. *International Journal of Human Computer Interaction* 59(1-2), 157–183 (2003)
18. Platt, J.C.: Sequential Minimal Optimizer: A Fast Algorithm for Training Support Vector Machines. Technical report, Microsoft Research (April 1998)
19. Scheirer, E., Slaney, M.: Construction and evaluation of a robust multifeature speech/music discriminator. In: Proceedings of ICASSP, vol. 1, pp. 1331–1334 (1997)
20. Scherer, K.R., Johnstone, T., Klasmeyer, G.: Affective Science. In: *Handbook of Affective Sciences - Vocal expression of emotion*, vol. 23, pp. 433–456. Oxford University Press, Oxford (2003)
21. Scherer, S., Schwenker, F., Palm, G.: Classifier fusion for emotion recognition from speech. In: Proceedings of Intelligent Environments 2007, pp. 152–155 (2007)
22. Schölkopf, B., Smola, A.J.: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge (2001)
23. Yacoub, S., Simske, S., Lin, X., Burns, J.: Recognition of emotions in interactive voice response systems. In: Proceedings of Eurospeech 2003 (2003)

A Simple Proof of the Convergence of the SMO Algorithm for Linearly Separable Problems

Jorge López and José R. Dorronsoro*

Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid, Spain

Abstract. We give a new proof of the convergence of the SMO algorithm for SVM training over linearly separable problems that partly builds on the one by Mitchell *et al.* for the convergence of the MDM algorithm to find the point of a convex set closest to the origin. Our proof relies in a simple derivation of SMO that we also present here and, while less general, it is considerably simpler than previous ones and yields algorithmic insights into the working of SMO.

1 Introduction

Given a sample $\mathcal{S} = \{(X_i, y_i) : i = 1, \dots, N\}$ with $y_i = \pm 1$, the standard formulation of SVM for linearly separable problems [1] wants to maximize the margin of a separating hyperplane (W, b) by solving the problem

$$\min \frac{1}{2} \|W\|^2 \text{ with } y_i(W \cdot X_i + b) \geq 1, \quad i = 1, \dots, N. \quad (1)$$

In practice, however, one solves the simpler dual problem of minimizing

$$f(\alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j X_i \cdot X_j - \sum_i \alpha_i \quad \text{with } \alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0. \quad (2)$$

There are quite a few proposals of algorithms to solve (2); many of them can be traced to two decomposition methods, Platt's SMO [2] or Joachims' SVM-Light [3] algorithms. When the SVM-Light working set has 2 patterns, it coincides with Keerthi *et al.*'s Modification 2 proposal for SMO (see [4]). Modification 2 [5] is by now the standard approach in first order SMO and from now on we will mean it when referring to SMO.

The convergence of SMO and, more generally, SVM-Light has been widely studied by several authors. The most comprehensive study has been that of Chih-Jen Lin and his coworkers who, in a series of papers, have proved the convergence of a general SVM-Light procedure even for non linearly separable problems under very general conditions on the kernel matrix. Specifically, asymptotic convergence of SVM-Light is proved in [6], whereas for the special case of SMO no assumption on the positive definiteness of the kernel matrix is needed in the proof [7]. Finally, in [8] the convergence of several

* With partial support of Spain's TIN 2007-66862 project and Cátedra UAM-IIC en Modelado y Predicción. The first author is kindly supported by FPU-MICINN grant AP2007-00142.

variants of SMO is proved, including the selection of the working set using second order information.

While general and powerful, these proofs are also involved and here we shall give an alternative, much simpler proof of the asymptotic convergence of SMO's Modification 2 for linearly separable problems that partly builds on a much earlier proof given in [9] of the convergence of the Mitchell–Demyanov–Malozemov algorithm (MDM) to find the point in a convex subset nearest to the origin. The closely related problem of finding the nearest points between two convex subsets has been long known to be equivalent to SVM solving [10] and, in fact, the generalized MDM algorithm that solves this second nearest point problem is closely related to Keerthi *et al.*'s Modification 2 [11].

Our proof relies in showing first that the distance between the current weight W and the optimal one W^* is dominated by the gap between the primal and dual problems and that, in turn, the gap is dominated by a quantity Δ that we call the violation extent and that is maximized by the selection of the indices U, L for the SMO working set. Essentially as done in [9], we will then show that some subsequence of the Δ values given by SMO tends to 0 and, hence, so do the corresponding W updates. At this point, a simple compactness argument ends the proof.

There is a clear limitation in the approach followed here, which is the fact that in its present form it only applies to linearly separable SVM training. On the other hand, our arguments reveal the working of the various SMO components towards the limit and gives some algorithmic insights on the overall behavior of SMO, such as, for instance, the fact that after a finite number of iterations, the only active patterns are those that lie in the support hyperplanes.

The paper is organized as follows. In section 2 we give a brief and simple overview of the SMO algorithm and pinpoint some facts that will be required for the convergence proof, which is given in section 3. Further insights on the behavior of SMO derived from the proof are given in section 4 and a brief discussion ends the paper.

2 A Simple Derivation of SMO for Linearly Separable Problems

We will briefly review here an approach that derives SMO from a maximum gain viewpoint (see also [11]). In general, SMO performs updates of the form $W' = W + \delta_L y_L X_L + \delta_U y_U X_U$, with δ_L, δ_U the update steps to be used. The restriction $\sum \alpha_i y_i = 0$ implies $\delta_U y_U = -\delta_L y_L$ and the updates become $W' = W + \delta y_L (X_L - X_U)$ where we write $\delta = \delta_L$ and, hence, $\delta_U = -y_U y_L \delta$. As a consequence, the multiplier updates are $\alpha'_L = \alpha_L + \delta$, $\alpha'_U = \alpha_U - y_U y_L \delta$ and $\alpha'_j = \alpha_j$ for other j . Therefore, the dual function $f(\alpha')$ on the new α' can be written as a function $\Phi(\delta)$ of the dilation factor δ , namely

$$\begin{aligned}\Phi(\delta) &= f(\alpha) + \delta y_L W \cdot (X_L - X_U) + \frac{1}{2} \delta^2 \|X_L - X_U\|^2 - \delta + y_L y_U \delta \\ &= f(\alpha) + \delta y_L W \cdot (X_L - X_U) + \frac{1}{2} \delta^2 \|X_L - X_U\|^2 - y_L \delta (y_L - y_U).\end{aligned}$$

Solving $0 = \Phi'(\delta) = y_L \{W \cdot (X_L - X_U) - (y_L - y_U)\} + \delta \|X_L - X_U\|^2$ yields the possibly optimal dilation factor

$$\delta = y_L \frac{W \cdot (X_U - X_L) - (y_U - y_L)}{\|X_L - X_U\|^2}$$

and implies that

$$f(\alpha') = f(\alpha) - \frac{(W \cdot (X_U - X_L) - (y_U - y_L))^2}{\|Z_{L,U}\|^2} = f(\alpha) - \frac{(\Delta_{U,L})^2}{\|Z_{L,U}\|^2},$$

where we write $\Delta_{U,L} = W \cdot (X_U - X_L) - (y_U - y_L)$ and $Z_{L,U} = X_L - X_U$. We may call $\Delta_{U,L}$ the violation extent associated to the U, L pair. Ignoring the $\|Z_{L,U}\|^2$ denominator, we can approximately maximize the gain in $f(\alpha')$ by choosing in principle L, U as

$$L = \arg \min \{W \cdot X_j - y_j\}, \quad U = \arg \max \{W \cdot X_j - y_j\}$$

so that the violation extent $\Delta_{U,L}$ is largest. Writing $\Delta = \Delta_{U,L}$ and $\lambda' = \frac{\Delta}{\|Z_{U,L}\|^2}$, we then have $\Delta > 0$, $\lambda' > 0$ and $\delta = y_L \lambda'$; moreover,

$$\Phi(\delta) = f(\alpha') = f(\alpha) - \lambda' \Delta + \frac{1}{2} (\lambda')^2 \|Z_{U,L}\|^2$$

and the α updates become $\alpha'_L = \alpha_L + y_L \lambda'$, $\alpha'_U = \alpha_U - y_U \lambda'$. Thus, α'_L or α'_U will decrease if $y_L = -1$ or $y_U = 1$. This requires the corresponding α_L and α_U to be > 0 and we must replace the previous L, U choices by

$$L = \arg \min \{W \cdot X_j - y_j : j \in \mathcal{I}_L\}, \quad U = \arg \max \{W \cdot X_j - y_j : j \in \mathcal{I}_U\},$$

where we use the notations $\mathcal{I}_U = \{i : (y_i = 1, \alpha_i > 0) \vee (y_i = -1)\}$ and $\mathcal{I}_L = \{i : (y_i = 1) \vee (y_i = -1, \alpha_i > 0)\}$. Moreover, to make sure that α'_L and α'_U remain then > 0 , we may have to clip λ' as follows

$$\lambda = \min\{\lambda', \alpha_L\} \text{ if } y_L = -1, \quad \lambda = \min\{\lambda', \alpha_U\} \text{ if } y_U = 1. \quad (3)$$

If λ is so clipped it turns out that either α'_L, α'_U or even both will become 0. Moreover, if, say, $\lambda = \alpha_L$ with $y_L = -1 = y_U$, we will then have $\alpha'_L = \alpha_L + y_L \lambda = 0$, $\alpha'_U = \alpha_U - y_U \lambda = \alpha_U + \alpha_L$. Thus, if $\alpha_U = 0$, it receives the previous value of α_L and the multiplier values are shuffled then among the X_i patterns and the number of non-zero multipliers does not change. If, however, α_U was already not zero, the new number of non-zero multipliers will decrease by one. A similar situation will hold when $\lambda = \alpha_U$ with $y_U = 1 = y_L$.

We finish this section with some observations to be used later on. First, notice that we clearly have $\lambda \|Z_{L,U}\|^2 \leq \Delta$ whatever the λ value chosen. As a consequence we will always have $f(\alpha') \leq f(\alpha) - \frac{1}{2} \lambda \Delta$; in particular, $f(\alpha') < f(\alpha)$ whenever $\Delta > 0$. On the other hand, if W, W' are feasible weight vectors (i.e., $y_i(W \cdot X_i + b) \geq 1$ for all i), we have

$$W \cdot W' = \sum \alpha_i y_i W' \cdot X_i = \sum \alpha_i y_i (W' \cdot X_i + b') \geq \sum \alpha_i.$$

In particular, if W^* denotes the optimal margin weight, $W \cdot W^* \geq \sum \alpha_i^* = \|W^*\|^2$, where the last equality follows from the KKT conditions for SMO.

3 The Convergence of SMO

We give now the proof of the convergence of SMO for linearly separable problems. As said above, it partly builds on the proof in [9] for the MDM algorithm; in fact, lemmas 3 and 4 are adapted from [9] while the other results in this section are specific for SMO.

Lemma 1. *We have $\|W - W^*\|^2 \leq \frac{1}{2}\Delta \sum \alpha_i$.*

Proof. The previous observations imply

$$\begin{aligned} \|W - W^*\|^2 &= \|W\|^2 - W \cdot W^* - (W \cdot W^* - \|W^*\|^2) \leq \|W\|^2 - W \cdot W^* \\ &\leq \|W\|^2 - \sum \alpha_i. \end{aligned}$$

In other words, $\|W - W^*\|^2$ is dominated by the dual gap, i.e., the difference between the primal and dual objective functions. Now, writing $I_{\pm} = \{i : y_i = \pm 1\}$ and observing that $\sum_{I_+} \alpha_i = \sum_{I_-} \alpha_j$, we have

$$\begin{aligned} \|W\|^2 - \sum \alpha_i &= \sum \alpha_i y_i W \cdot X_i - \sum \alpha_i y_i^2 = \sum \alpha_i y_i (W \cdot X_i - y_i) \\ &= \sum_{I_+} \alpha_i (W \cdot X_i - y_i) - \sum_{I_-} \alpha_i (W \cdot X_i - y_i) \\ &\leq \max_{\{I_+, \alpha_i > 0\}} \{W \cdot X_i - y_i\} \sum_{I_+} \alpha_i - \min_{\{I_-, \alpha_i > 0\}} \{W \cdot X_i - y_i\} \sum_{I_-} \alpha_i \\ &= \frac{1}{2} \left(\max_{\{I_+, \alpha_i > 0\}} \{W \cdot X_i - y_i\} - \min_{\{I_-, \alpha_i > 0\}} \{W \cdot X_i - y_i\} \right) \sum \alpha_i \\ &\leq \frac{1}{2} \left(\max_{\mathcal{I}_U} \{W \cdot X_i - y_i\} - \min_{\mathcal{I}_L} \{W \cdot X_i - y_i\} \right) \sum \alpha_i \\ &= \frac{1}{2} \Delta \sum \alpha_i, \end{aligned}$$

as we wanted to prove. \square

Starting with some feasible W^0 , SMO iteratively applies the previous index selection and multiplier updates to build a sequence of multipliers α^k and weight vectors W^k . The null vector $W^0 = 0$ is clearly feasible and $f(\alpha^0) = 0$. Thus, we will have $f(\alpha^k) \leq 0$ for all k , i.e., $\|W^k\|^2 \leq 2 \sum \alpha_i^k$.

Lemma 2. *If $f(\alpha^0) \leq 0$, the sums $\sum_i \alpha_i^k$ of the successive SMO multipliers will be bounded.*

Proof. Cauchy's inequality and the fact that we assume the matrix $Q = (Q_{ij})$, with $Q_{ij} = y_i y_j X_i \cdot X_j$, to be positive definite imply

$$\sum_1^N \alpha_i^k \leq \sqrt{N} \left(\sum_1^N (\alpha_i^k)^2 \right)^{\frac{1}{2}} \leq \sqrt{N} \eta \|W^k\|.$$

This and the preceding observation yield

$$\|W^k\|^2 \leq 2 \sum_i \alpha_i^k \leq 2\sqrt{N}\eta\|W^k\|,$$

and, therefore, $\|W^k\| \leq 2\sqrt{N}\eta$. Thus, using this and the feasibility conditions at each X_i , we arrive at $\sum_1^N \alpha_i^k \leq \sum_1^N \alpha_i^k y_i(W^k \cdot X_i + b^k) = W^k \cdot W^k = \|W^k\|^2 \leq 4N\eta^2$. \square

Let λ_k and Δ_k be the λ and Δ values obtained in the SMO iterations. We show next that for some subsequence k_j , we have $\Delta_{k_j} \rightarrow 0$, which, by lemmas (1) and (2), will also imply $W^{k_j} \rightarrow W^*$. We first have

Lemma 3. $\lim \lambda_k \Delta_k = 0$.

Proof. Assume to the contrary that $\lambda_k \Delta_k$ does not converge to 0. There is then an $\epsilon > 0$ and a subsequence k_j such that $\lambda_{k_j} \Delta_{k_j} > 2\epsilon$. Now, as observed above, we would then have

$$f(\alpha^{1+k_j}) \leq f(\alpha^{k_j}) - \frac{1}{2}\lambda_{k_j} \Delta_{k_j} < f(\alpha^{k_j}) - \epsilon,$$

and since the sequence $f(\alpha^k)$ decreases, we would have

$$\begin{aligned} f(\alpha^{1+k_{j+M}}) &< f(\alpha^{k_{j+M}}) - \epsilon \leq f(\alpha^{1+k_{j+M}-1}) - \epsilon < f(\alpha^{k_{j+M}-1}) - 2\epsilon \leq \dots \\ &\leq f(\alpha^{1+k_j}) - M\epsilon < f(\alpha^{k_j}) - (M+1)\epsilon, \end{aligned}$$

which would eventually contradict that $f(\alpha^k) \geq f(\alpha^*)$ for all k . \square

The inferior limit of a sequence a_n is $\liminf a_n = \sup_n \inf_{m \geq n} \{a_m\} = \lim(\inf_{m \geq n} \{a_m\})$. We have now

Lemma 4. $\liminf \Delta_k = 0$.

Proof. If not true, there is some $\epsilon > 0$ such that $\Delta_k \geq \epsilon$. Then by the previous lemma, $\lambda_k \rightarrow 0$ and we will show that this leads to a subsequence k_j for which $\Delta_{k_j} \rightarrow 0$, contradicting our starting assumption on the Δ_k . To do so, we claim that there must be a subsequence k_j for which

$$\lambda_{k_j} = \lambda'_{k_j} = \frac{\Delta_{k_j}}{\|X_{L_{k_j}} - X_{U_{k_j}}\|^2}, \quad (4)$$

i.e., no λ clipping takes place. If so, writing $R = \max\{\|X_m\|\}$, we have $\|X_i - X_j\|^2 \leq 2 \max\{\|X_m\|^2\} = 2R^2$, and, therefore, $\Delta_{k_j} = \lambda_{k_j} \|X_{L_{k_j}} - X_{U_{k_j}}\|^2 \leq 2R^2 \lambda_{k_j}$. Thus, $\Delta_{k_j} \rightarrow 0$ as we wanted to show.

It remains to prove (4). Recall that when the λ_k are not defined as in (4), we must have either $\lambda_k = \alpha_L^k$ or $\lambda_k = \alpha_U^k$. To obtain the desired subsequence we show that the number of different subsequent consecutive updates for which λ_{k+m} equals α_L^{k+m} or α_U^{k+m} must be a finite M . In fact, let N_k denote the number of non-zero multipliers in α^k and assume that clipped updates are applied at each iteration. Then, as observed

before, for any such update p we will clearly have $N_{p+1} \leq N_p$. Thus, if these clipped updates would go on indefinitely, N_p should become constant and > 0 at some iteration q (it cannot become zero). But then, the subsequent updates would simply interchange the values of the α^q multipliers among the sample patterns, which after a finite number of iterations would lead to a pair (r, s) such that $W^r = W^s$ for $s > r \geq q$, contradicting that $f(\alpha^s) < f(\alpha^r)$. Hence, the clipped updates should stop at some point and there must be a subsequence k_j verifying (4). \square

Now a general compactness argument ends the convergence proof.

Theorem 1. *The sequence W^k provided by SMO verifies $W^k \rightarrow W^*$.*

Proof. As a consequence of the previous lemmas, we know that there is a subsequence W^{k_j} for which $W^{k_j} \rightarrow W^* = \sum \alpha_i^* y_i X_i$. Moreover, the SMO algorithm ensures that $f(\alpha^{k+1}) \leq f(\alpha^k)$ and, therefore, $f(\alpha^k) \rightarrow f(\alpha^*)$. Assume that, nevertheless, W^k does not converge to W^* . For some ϵ there is then a sequence W^{p_q} such that $\|W^{p_q} - W^*\|^2 \geq \epsilon$. Since the W^k form a bounded subset, there is a subsequence $W^{p_{q_m}}$ such that $W^{p_{q_m}} \rightarrow W'$ for some W' . Then we must have $f(\alpha') = f(\alpha^*)$ and since all the W^k are feasible, so is W' . But by uniqueness of the primal solution [2], it would then follow that $W' = W^*$ and this contradiction implies that $W^k \rightarrow W^*$. \square

4 Other Consequences

We first observe that $\sum_i \alpha_i^k \rightarrow \sum_i \alpha_i^*$. In fact, by theorem I $\|W^k\|^2 \rightarrow \|W^*\|^2$ and we have then

$$\sum_i \alpha_i^k = \frac{1}{2} \|W^k\|^2 - f(\alpha^k) \rightarrow \frac{1}{2} \|W^*\|^2 - f(\alpha^*) = \sum_i \alpha_i^*.$$

Next, the KKT conditions for SVM imply that at the optimal α^* , W^* and b^* we must have $\alpha_i^* [y_i(W^* X_i + b^*) - 1] = 0$. Let \mathcal{H}_1 denote the index set of those i such that $y_i(W^* X_i + b^*) = 1$ and \mathcal{H}_2 denote the index set of those i such that $y_i(W^* X_i + b^*) > 1$. The main result of this section, Theorem 2, states that after a finite number of iterations all the non-zero multipliers in W^k must be taken from \mathcal{H}_1 . In other words, if $S_k = \{i : \alpha_i^k > 0\}$, then $S_k \subset \mathcal{H}_1$. We shall prove this after two lemmas, which we precede by the following observations.

First, it is clear from the KKT conditions that $\{i : \alpha_i^* > 0\} \subset \mathcal{H}_1$ and if $i \in \mathcal{H}_1$ we have $W^* X_i - y_i = -b^*$. Set $\rho = \min\{y_i(W^* X_i + b^*) - 1 : i \in \mathcal{H}_2\}$. Then, if $i \in \mathcal{H}_2 \cap I_+$ we have $W^* X_i + b^* \geq 1 + \rho = y_i + \rho$ and $W^* X_i - y_i \geq -b^* + \rho$; similarly, if $i \in \mathcal{H}_2$ and $y_i = -1$, $W^* X_i + b^* \leq -1 - \rho = y_i - \rho$ and $W^* X_i - y_i \leq -b^* - \rho$. On the other hand, if $R = \max_i \|X_i\|$, since

$$\max |y_i W^k \cdot X_i - y_i W^* \cdot X_i| \leq \|W^k - W^*\| R$$

and $W^k \rightarrow W^*$, we must also have $\max |y_i W^k \cdot X_i - y_i W^* \cdot X_i| \rightarrow 0$.

Lemma 5. *There is a K_1 such that if $k \geq K_1$,*

$$\begin{aligned} W^k \cdot X_i - y_i &\geq \frac{\rho}{2} + W^k \cdot X_j - y_j \text{ for all } i \in \mathcal{H}_2 \cap I_+, j \in \mathcal{H}_1; \\ W^k \cdot X_i - y_i &\leq -\frac{\rho}{2} + W^k \cdot X_j - y_j \text{ for all } i \in \mathcal{H}_2 \cap I_-, j \in \mathcal{H}_1. \end{aligned}$$

Proof. Let K_1 be such that for $k \geq K_1$, $\max_{i \in \mathcal{H}_2 \cap I_+, j \in \mathcal{H}_1} |y_i W^k \cdot X_i - y_i W^* \cdot X_i| \leq \rho/4$. If $i \in \mathcal{H}_2 \cap I_+, j \in \mathcal{H}_1$, we have

$$\begin{aligned} W^k \cdot X_i - y_i &= W^k \cdot X_i - W^* \cdot X_i + W^* \cdot X_i - y_i \geq -\frac{\rho}{4} - b^* + \rho \\ &= \frac{3\rho}{4} + W^* \cdot X_j - y_j = \frac{3\rho}{4} + W^* \cdot X_j - W^k \cdot X_j + W^k \cdot X_j - y_j \\ &\geq \frac{3\rho}{4} - \frac{\rho}{4} + W^k \cdot X_j - y_j = \frac{\rho}{2} + W^k \cdot X_j - y_j, \end{aligned}$$

and the same argument yields $W^k \cdot X_i - y_i \leq -\frac{\rho}{2} + W^k \cdot X_j - y_j$ for $i \in \mathcal{H}_2 \cap I_-, j \in \mathcal{H}_1$. \square

As a consequence, notice that if $\mathcal{H}_2 \cap I_+ \cap S_k \neq \emptyset$, we shall have $U_k \in \mathcal{H}_2$. Similarly, if $\mathcal{H}_2 \cap I_- \cap S_k \neq \emptyset$, we shall have $L_k \in \mathcal{H}_2$. On the other hand, if $\mathcal{H}_2 \cap I_+ \cap S_k = \emptyset$, we will take $U_k \in \mathcal{H}_1$ and we shall also have $L_k \in \mathcal{H}_1$ if $\mathcal{H}_2 \cap I_- \cap S_k = \emptyset$.

Another consequence is that for $k \geq K_1$, if either $\mathcal{H}_2 \cap I_+ \cap S_k \neq \emptyset$ or $\mathcal{H}_2 \cap I_- \cap S_k \neq \emptyset$, we will have $\Delta^k \geq \rho/2$.

Lemma 6. *The quantity $\sum_{\mathcal{H}_2} \alpha_i^k$ tends to 0 as $k \rightarrow \infty$.*

Proof. We can decompose any W^k in the SMO sequence as $W^k = W_1^k + W_2^k$, with

$$W_1^k = \sum_{\mathcal{H}_1} \alpha_i^k y_i X_i, \quad W_2^k = \sum_{\mathcal{H}_2} \alpha_j^k y_j X_j.$$

Therefore,

$$\begin{aligned} (W^k - W^*) \cdot W^* &= \sum \alpha_i^k y_i (W^* \cdot X_i + b^*) - \sum \alpha_j^k y_j (W^* \cdot X_j + b^*) \\ &= \left(\sum_{\mathcal{H}_1} + \sum_{\mathcal{H}_2} \right) \alpha_i^k y_i (W^* \cdot X_i + b^*) - \sum \alpha_i^* \\ &\geq \sum_{\mathcal{H}_1} \alpha_i^k + (1 + \rho) \sum_{\mathcal{H}_2} \alpha_i^k - \sum \alpha_i^* \\ &= \sum_{\mathcal{H}_2} \alpha_i^k - \sum \alpha_i^* + \rho \sum_{\mathcal{H}_2} \alpha_i^k, \end{aligned}$$

and it follows that $\sum_{\mathcal{H}_2} \alpha_i^k \rightarrow 0$. \square

We turn now to the desired result.

Theorem 2. *There is a K_0 such that for all $k \geq K_0$, we have $S_k \subset \mathcal{H}_1$; that is, all the non-zero multiplier vectors in W^k will be in the optimal support hyperplanes.*

Proof. We know that there is a K_1 such that for all $k \geq K_1$, $\Delta^k \geq \rho/2$ if $S_k \cap \mathcal{H}_2 \neq \emptyset$ and, as a consequence, $\frac{\Delta^k}{\|Z\|^2} \geq \frac{\rho}{2R^2}$. Moreover, we can find a K_2 such that for all $k \geq K_2$, $\sum_{\mathcal{H}_2} \alpha_i^k \leq \frac{\rho}{2R^2}$. Thus, if $K_3 = \max(K_1, K_2)$ and either L_k or U_k are taken from \mathcal{H}_2 , it will follow that λ_k will be either $\alpha_{U_k}^k$ or $\alpha_{L_k}^k$ whenever $S_k \cap \mathcal{H}_2 \neq \emptyset$ for $k \geq K_3$. In other words, at least one pattern X_j with $j \in \mathcal{H}_2$ will be removed from the representation of W^{k+1} as its multiplier is set to 0. Moreover, if a new pattern X_i is added to W^{k+1} (i.e., $\alpha_i^k = 0$, but $\alpha_i^{k+1} > 0$), we must have $i \in \mathcal{H}_1$. It is thus clear that in at most $|\mathcal{H}_2|$ iterations after K_3 , all the \mathcal{H}_2 multipliers are zero and are no longer considered when computing U_k and L_k for $k \geq K_0 = K_3 + |\mathcal{H}_2|$. \square

Finally, since the U_k and L_k indices in $\Delta^k = \Delta_{U_k, L_k}^k$ must be taken from \mathcal{H}_1 when $k \geq K_0$, it follows that Δ^k and, hence, λ_k tend to 0.

5 Conclusions and Further Work

In this work we present a new, simple and direct proof of the convergence of Keerthi *et al.*'s Modification 2 for the SMO algorithm when applied to linearly separable problems which partly builds on the proof of the MDM algorithm for the minimum norm problem given in [9]. Many authors, most notably C.J. Lin and his coworkers, have exhaustively studied the convergence of several general decomposition methods for SVM classification and regression that include SVM Light, SMO and various first and second order variants. While much less general than these results, our proof relies on the concrete steps of the SMO algorithm. In particular, it is simpler and more direct than other convergence proofs and sheds light on some aspects of the algorithm, such as how the dual gap controls the distance between the current weight vector and the optimal one or how this gap is, in turn, dominated by the violation extent value Δ that SMO uses to select the new pair of updating patterns. We show that Δ tends to zero and that, after a finite number of iterations, all the updating pattern pairs will be taken from the support hyperplanes. Our proof implies the convergence of SMO for quadratic penalty SVM problems, as they can be written as linearly separable SVM problems in an extended weight and pattern space [13]. Also, although not proved here, our arguments apply to Support Vector Regression.

As of future work, an obvious step is to extend the proof here to SMO for non-linearly separable problems. However, a more interesting approach would be to further exploit the arguments here to obtain other insights on the behavior of SMO that may help in devising more efficient SVM training algorithms. We are currently working on these and other related questions.

References

1. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
2. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods - Support Vector Machines, pp. 185–208 (1999)
3. Joachims, T.: Making large-scale support vector machine learning practical. In: Advances in Kernel Methods - Support Vector Machines, pp. 169–184 (1999)

4. Fan, R.-E., Chen, P.-H., Lin, C.-J.: Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research* 6, 1889–1918 (2005)
5. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation* 13(3), 637–649 (2001)
6. Lin, C.-J.: On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks* 12(6), 1288–1298 (2001)
7. Lin, C.-J.: Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks* 13(1), 248–250 (2002)
8. Chen, P.-H., Fan, R.-E., Lin, C.-J.: A study on SMO-type decomposition methods for support vector machines. *IEEE Transactions on Neural Networks* 17, 893–908 (2006)
9. Mitchell, B.F., Dem'yanov, V.F., Malozemov, V.N.: Finding the point of a polyhedron closest to the origin. *SIAM J. Contr.* 12, 19–26 (1974)
10. Bennett, K.P., Bredensteiner, E.J.: Duality and geometry in SVM classifiers. In: *Proceedings of the 17th International Conference on Machine Learning*, pp. 57–64 (2000)
11. López, J., Barbero, Á., Dorronsoro, J.: On the equivalence of the SMO and MDM algorithms for SVM training. In: *Machine Learning and Knowledge Discovery in Databases - ECML 2008 Proceedings, Part II. LNCS (LNAI)*. Springer, Heidelberg (2008)
12. Burges, C.J.C., Crisp, D.J.: Uniqueness theorems for kernel methods. *Neurocomputing* 55(1-2), 187–220 (2003)
13. Shawe-Taylor, J., Cristianini, N.: On the generalization of soft margin algorithms. *IEEE Transactions on Information Theory* 48(10), 2721–2735 (2002)

Spanning SVM Tree for Personalized Transductive Learning

Shaoning Pang¹, Tao Ban², Youki Kadobayashi², and Nik Kasabov¹

¹ Knowledge Engineering & Discover Research Institute

Auckland University of Technology, Private Bag 92006, Auckland 1020, New Zealand

{sn pang, nkasabov}@aut.ac.nz

² Information Security Research Center, National Institute of Information and Communications Technology, Tokyo, 184-8795 Japan
bantao@nict.go.jp, youki-k@is.aist-nara.ac.jp

Abstract. Personalized Transductive Learning (PTL) builds a unique local model for classification of each test sample and therefore is practically neighborhood dependant. While existing PTL methods usually define the neighborhood by a predefined (dis)similarity measure, in this paper we introduce a new concept of knowledgeable neighborhood and a transductive SVM classification tree (t-SVMT) for PTL. The neighborhood of a test sample is constructed over the classification knowledge modelled by regional SVMs, and a set of such SVMs adjacent to the test sample are aggregated systematically into a t-SVMT. Compared to a regular SVM and other SVMTs, the proposed t-SVMT, by virtue of the aggregation of SVMs, has an inherent superiority on classifying class-imbalanced datasets. Furthermore, t-SVMT has solved the over-fitting problem of all previous SVMTs as it aggregates neighborhood knowledge and thus significantly reduces the size of the SVM tree.

1 Introduction

The widely used inductive reasoning approach is concerned with the creation of a model from all training data which represents the available information from the problem space (induction), and then the application of the model to new coming data to predict the property of interest (deduction or generalization). In contrast, a transductive approach, first introduced by Vapnik [1], is defined as a method which estimates the value of an unknown model for a single point in the problem space (a sample vector) employing information relevant to that vector. While the inductive approach is useful when a global model of the problem is needed in an approximate form, the transductive approach is more appropriate for applications where the focus is not on the overall precision of the model, but rather on every individual case. In this sense, transductive learning well fits the cases in clinical and medical applications where the interest is on the preciseness of the prediction for an individual patient.

Different from an inductive learner which conducts learning only on training data $h_L = L_i(S_{train})$, a transductive learner learns from both training and

test data, $h_L = L_t(S_{train}, S_{test})$. With respect to how the unlabelled data S_{test} can be used for learning, personalized transductive learning methods approach ‘personalized’ learning by creating a unique model for each test sample based on its neighboring samples. A typical personalized learning approach usually consists of two steps: (1) *Neighbor-sample filtering*: for each new input sample $\mathbf{x}_j \in S_{test}$ that needs to be processed for a prognostic/classification task, its N_j nearest neighbors are selected from the training set to form a neighborhood set $D_j \subseteq S_{train}$. (2) *Regional decision making*: a personalized model M_j is created on D_j to approximate the function value y_j at point \mathbf{x}_j .

The simplest personalized transductive model is the k nearest neighbor classifier (k NN). k NN dynamically creates a prediction model for each test sample by learning upon its k nearest neighbors. Neuro-Fuzzy Inference (NFI) is a newly introduced personalized transductive learning method [2]. By employing a fuzzy inference model for regional decision making, NFI outperforms k NN on some benchmark data sets. However, since the personalized prediction models for NFI and k NN are based on the same neighbor-sample filtering, their generalization abilities are not significantly different.

It is worth noting that for data collected from real world applications, the neighborhood directly defined by inter-sample dissimilarities is subjected to high noise or ill-posed conditions, which in turn renders the personalized prediction models unreliable. A neighborhood based on inter-sample dissimilarity is unable to approximate such neighborhood data distribution precisely, because samples adjacent to the query sample might be just noise data. Thus for a better neighborhood modelling, it is desirable to take into consideration, besides the distance metric, also the classification information/knowledge to enable constructed neighborhood knowledgeable. Motivated by this, we propose in this paper a transductive SVM tree (t-SVMT), which implements personalized multi-model cooperative learning in a transductive manner.

2 Personalized Neighborhood Calculation

Given a query sample \mathbf{x} in data space D , the neighborhood of \mathbf{x} is defined as a subset Z_x adjacent to \mathbf{x} , possibly containing samples from different classes.

$$Z_x = \{\mathbf{z}_1, \dots, \mathbf{z}_k, \mathbf{x} \mid d(\mathbf{z}_i, \mathbf{x}) < \theta\}, \quad (1)$$

where \mathbf{z}_i represents a neighboring sample, $d(\mathbf{z}_i, \mathbf{x})$ is the distance between neighboring sample and the query sample \mathbf{x} , θ is a predefined neighborhood up bound.

Personalized transductive learning assumes that $f(\mathbf{x}, Z_x)$ approximates the ground truth $f(\mathbf{x})$ better than $f(\mathbf{x}, D)$ for the reason that noise data is removed due to neighboring instance selection. Thus, a personalized transductive function can be learned just from the neighborhood data set Z_x , but not from the full data set D . However, it is noticeable that (I) exclusively relies on inter-sample dissimilarity evaluation. The obtained neighborhood, depending on what kind of dissimilarity measure is used, is presented either as a circle-type scope shown in Fig. I (b) or as a rectangle of Fig. I (c). Such neighborhood calculation is

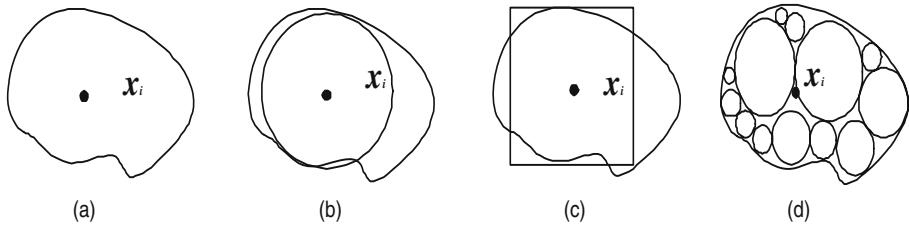


Fig. 1. Illustration of dynamic neighborhood modelling. (a) Truth personalized neighborhood of x_i ; (b) circle-type neighborhood approximation; (c) rectangle-type neighborhood approximation; (d) the proposed subset aggregated neighborhood.

improper when a dynamic neighborhood is required for more accurate personalized modelling. Moreover, the neighborhood obtained from (II) is merely in the sense of dissimilarity measure, ignoring discriminative knowledge between different classes, which in some cases leads to degeneration in the generalization performance.

A personalized neighborhood Z_x is modelled based on the following two policies: (1) class label information is incorporated for defining the neighborhood, and (2) a neighborhood is composed of flexible-size multi-instance packages instead of individual instances. Mathematically, the proposed personalized neighborhood model is defined as

$$Z_x^* = Z_1 \cup Z_2 \cup \dots \cup Z_k \cup \{x\}, \text{ Subject to } d(Z_i, x) < \theta, \text{ and } Z_i \leftarrow \wp(D), \quad (2)$$

where Z_i is a subset containing neighboring samples to x , $d(Z_i, x)$ the distance between Z_i and query sample x , and θ a predefined dissimilarity upper bound. It is noticeable that Z_i is a flexible-size multi-instance package, which is a subset of D , or sometimes only a single instance. Also, Z_x^* considers the class label information as every Z_i is obtained from a supervised clustering $\wp(D)$. Therefore, Z_x^* is personalized for x in terms of the membership of the neighborhood. As an illustration, Fig. II(d) gives an aggregated neighborhood by a set of multi-instance packages represented as circles in the figure. It is obvious that Fig. II(d) presents a more accurate neighborhood approximation than Fig. II(b) and (c).

3 Personalized Mode Transductive Spanning SVM Tree

SVM tree (SVMT) [4] is a type of inductive SVM aggregation. It overcomes the difficulty of SVM ensemble by determining the number of SVMs automatically during learning. The disadvantage of existing SVMTs [34] is that the spanning of SVMT is completely data-driven (called DDS-SVMT here afterward): the tree grows easily into a large-size SVM decision tree with terminal SVM nodes built on extremely small dataset, which results in the generated SVMT over-fitting to the training data. Here we present a new SVM aggregating method which constructs an SVMT in a transductive way. The proposed approach is capable of

preventing the over-fitting of SVMT learning as it aggregates only neighborhood knowledge and hence largely reduces the size of generated SVM tree. Towards a multi-model transductive SVM aggregation classifier with high generalization ability, in this section we address three aspects of the model: (1) problem decomposition, i.e. data partitioning, (2) modelling of local knowledge, i.e. individual SVM training on regional multi-instance package, and (3) individual SVMs aggregation, i.e. transductive aggregation.

3.1 Data Partitioning

In the spirit of searching for better partitions with good discriminative ability, we are aiming at partitioning data into geographically adjacent subsets while taking class label information into account.

Let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ be the training set, and the associative labels given by $\{y_1, \dots, y_N\}$, a new attribute z_i can be formed out of the original attributes as $z_i = (\mathbf{x}_i, y_i)$, embodying the information contained in X as a set of data partitions $\{Z_1, \dots, Z_k\}$. A supervised data decomposition can be accomplished by introducing cost function based on a regional SVM approximation,

$$E(Z_k) = \sum_{x \in Z_k} (\mathbf{x} - \mathbf{x}_k) + \alpha \sum_{x \in Z_k} (f_{svm}(\mathbf{x}_k) - f_{svm}(\mathbf{x})), \quad (3)$$

where f_{svm} is a standard SVM approximation given later, and α is a balancing weight.

Given a partitioning function \wp on X with adjustable partitioning scale, (3) can be used for a supervised recursive partitioning procedure as,

$$[Z_1, \dots, Z_i, \dots] = \wp^n(X, \rho_0), \text{ Subject to } E(Z_i) < \xi, \quad (4)$$

where ρ is the partitioning scale, ξ the threshold of E , and Z_1, \dots, Z_k are the set of partitions selected by optimizing the cost E . $\wp^n = \wp(\wp^{n-1}(X_{n-1}, \rho_{n-1}), \rho_n)$, $\rho_n = \rho_{n-1} + \Delta$, and Δ is the partitioning scale interval normally set to a small value in the range of \wp definition. For example, for K-means, Δ can be an integer greater-equal to 1.

By (4), X can be partitioned into a set of data partitions in different sizes $\{Z_1, Z_2, \dots, Z_i, \dots\}$, such that $Z_1 \cup Z_2 \cup \dots \cup Z_i \cup \dots \approx D$, and each partition Z_i is associated with the partitioning scale ρ , in which Z_i is produced.

3.2 SVM Particles

To formalize classification as the task of finding partitions with maximum likelihood, a local SVM is associated with each partition obtained by the above data partitioning procedure. An *SVM particle* is defined as a structure that combines the dataset Z_i , and the trained SVM f_{svm}^i on Z_i , $V_i = \{Z_i, f_{svm}^i\}$.

In the case that Z_i has data from two classes, a regular two-class SVM model $f_{svm} = sign(\sum_{i=1}^l y_i (\mathbf{w}_i^T \varphi(\mathbf{x}_i) + b_i^*))$, where φ is the kernel function, l is the number of training samples, and \mathbf{w}_i and b_i^* are optimized by

$$\min \frac{1}{2} \mathbf{w}_i^T \mathbf{w}_i + C \sum_{t=1}^l \eta_i^2, \text{ Subject to } y_i(\mathbf{w}_i^T \varphi(\mathbf{x}_t) + b_i) \geq 1 - \eta_i, \quad (5)$$

where C is the margin parameter to weight the error penalties η_i , and $\varphi(\cdot)$ is the mapping function implemented by the kernel function. On the other hand, when Z_i only contains data from one class, a one-class SVM is applied. Following [5], a class SVM function can be modelled as an outlier classifier by labelling samples in Z_i as positive and samples in the complement set \overline{Z}_i as negative. Then the SVM is trained on a dataset $Z'_i = \{\mathbf{x}_i | i = 1, \dots, N\}$, and

$$y'_i = \begin{cases} +1 & \text{if } \mathbf{x}_i \in Z_i, \\ -1 & \text{if } \mathbf{x}_i \in \overline{Z}_i. \end{cases} \quad (6)$$

Consequently, the following two types of SVM particles are obtained: (1) one-class SVM particles $V^{<1>} = \{g, \rho, f_{svm<1>}\}$ for class 1, $V^{<2>} = \{g, \rho, f_{svm<2>}\}$ for class 2, and (2) two-class SVM particle $V^{<2>} = \{g, \rho, f_{svm}\}$. Given a data partition belonging to one class, a one-class SVM is applied to model the data by separating the samples in the partition from the outliers. In the case that the data belong to two classes, satisfying $E(g) > \xi$, then a two-class SVM is applied to the partition, and a standard SVM is created.

3.3 Personalized Mode Transductive Aggregation

Given a test sample \mathbf{x} , and a set of SVM particles $\{V_i\}$ derived over S_{train} , the distance between \mathbf{x} and a particle Z_i is measured by the normalized Euclidean distance defined as follows,

$$d(\mathbf{x}, Z_i)^2 = (\mathbf{x} - \frac{\sum_{\mathbf{x}_k \in Z_i} \mathbf{x}_k}{|Z_i|})^T (\mathbf{x} - \frac{\sum_{\mathbf{x}_k \in Z_i} \mathbf{x}_k}{|Z_i|}), \quad (7)$$

where $|Z_i|$ denotes the cardinal number of the particle.

Then, all selected SVM particles compose the knowledgeable neighborhood for the new input instance \mathbf{x} , and the decision of classification is made transductively by an aggregation \hat{f} of those SVM particles in the neighborhood as,

$$\hat{f}(\mathbf{x}) = \begin{cases} f_{svm<1>} & \text{if } \mathbf{x} \rightharpoonup V^{<1>} \\ f_{svm<2>} & \text{if } \mathbf{x} \rightharpoonup V^{<2>} \\ f_{svm} & \text{otherwise,} \end{cases} \quad (8)$$

where $f_{svm<1>}$ and $f_{svm<2>}$ are one-class SVM decision makers for class 1 and 2, respectively, and f_{svm} is a 2-class SVM decision maker. Taking each SVM particle as a tree node and the partitioning scale ρ as the level of tree structure, \hat{f} can be represented as an SVM tree. Thus, ‘ \rightharpoonup ’ in (8) indicates that \mathbf{x} is branched to a certain node of the SVM tree.

Clearly, there may be errors in the classification of above constructed \hat{f} , as \hat{f} may differ from the true classification function f . Thus, a suitably chosen real-value loss function $\mathcal{L} = \mathcal{L}(\hat{f}, f)$ is used to capture the extent of this error.

Therefore loss \mathcal{L} is data dependent: $\mathcal{L} = |\hat{f} - y|$. As \hat{f} is applied to datasets drawn from the data domain D under a distribution of $\theta(\mathbf{x})$. The expected loss can be quantified as,

$$\mathcal{L} = \sum_{i=1}^I |f_{svm_i} - f_i| \varrho_i + \sum_{j=1}^J |f_{svm_j^{<1>}} - f_j| q_{1j} + \sum_{k=1}^K |f_{svm_k^{<2>}} - f_k| q_{2k}, \quad (9)$$

where f_i , f_j , and f_k are the regional true classification function. ϱ_i is the probability that the i th particle contains two class data. $q_r(r = 1, 2)$ represent the probability that a partition contains data from only one class. I , J , and K represent the number of two-class SVM particles, one-class SVM particles for class 1, and class 2, respectively. Here, I , J , K and N are determined automatically after \hat{f} is created.

From (7), the aggregation risk is determined by 2-class SVM and one-class SVM classifiers over all the particles/regions. Also, the risk from ϕ has already been minimized during the process of data partitioning. Thus an optimized SVM aggregation is achieved as long as the risk from every regional SVM is minimized.

4 The Proposed t-SVMT Algorithm

In this section, the above personalized transductive modelling is interpolated as an algorithm of transductive SVM tree (t-SVMT). First, the training data is divided in a recursive data partitioning procedure. Then, regional knowledge of the input data is approximated as a set of SVM particles. Finally, the selected transductive particles (i. e. particles neighboring to a test instance \mathbf{x}) are aggregated into a personalized SVM tree model. Below is a step by step explanation of the operations taken by the t-SVMT Algorithm:

Step 1: Partitioning. The input training data is decomposed and modelled into a set of SVM particles. Algorithm 1 gives the *partitioning* function, where ρ_0 is a predefined initial resolution for t-SVMT to start analyzing. Default ρ_0 is normally set as a scale that gives the biggest size data partitions. For example, for K-mean, ρ_0 is set as 2. If some prior knowledge of the data is known, e.g. serious class-imbalance and class-overlap, a finer scale is suggested to enable t-SVMT to analyze the data with a finer resolution. In our experiments, we adopt a standard K-mean clustering approach with $\rho = 2$.

Step 2: Spanning transductive SVMT. Given a new input instance \mathbf{x} , a personalized t-SVMT \mathcal{T}_i is constructed by transductive SVM particle training followed by SVMT aggregation. Algorithm 2 describes the *t-SVMTtraining* function. In our experiments, two-class SVMs use a linear kernel and one-class SVMs employ RBF kernel with parameter adjusted via cross-validation.

Step 3: Testing the constructed t-SVMT $y = \mathcal{T}(\mathbf{x})$. Test sample \mathbf{x} is first judged by the test function $\mathcal{T}_0(x)$ at the root node in the SVM tree. Depending on the decision made by the root node, \mathbf{x} will be branched to one of the children of the root node. This procedure is repeated until a leaf node or an SVM node is

```

Function: Partitioning( $X_{train}, \rho_0$ )
 $X_{train}$ ; /* training dataset */
 $\rho$ ; /* initial partition scale */
 $P$ ; /* output data partition set*/
begin
1  $P = \emptyset$ ; /* initialize output */
2  $\rho = \rho_0$ ; /* initialize partition scale*/
3 if  $X_{train}$  is empty
4   return  $P$ ; /* Iteration stops when  $X_{train}$  is empty */
5  $[Z_1, \dots, Z_k] = \text{Partition}(X_{train}, \rho_0)$ ;
6 for each  $Z_k$  {
7   if all  $x \in Z_k$  is in one-class
8      $P = P \cup [Z_k, \rho]$ ;
9   if  $Z_k$  in two-class, and  $E(Z_k) > \xi$ 
10     $P = P \cup [Z_k, \rho]$ ;
11    $X_{train} = X_{train} - Z_k$ ; }
12 if  $X_{train}$  size is not decreasing
13    $\rho = \rho - \Delta$ ;
14 Partitioning( $X_{train}, \rho$ );/* zooming in */
end

```

Algorithm 1. Partitioning

reached, then the final classification decision is made for the test sample by a node one-class classifier or a regular SVM classification.

5 Experiments and Discussions

In our experiments, we compare the results of t-SVMT with those of standard inductive SVM, and previous SVM aggregation methods: SVM ensemble [7] and DDS_SVMT [4]. On the other hand, we also compare t-SVMT with other transductive methods including transductive SVM [6], kNN and NFI [2]. In the comparison, we set all SVMs with a linear kernel, and aggregation parameters for SVMTs (e.g. ρ_0 and ξ) as well as parameters for kNN and NFI to be determined by cross validation experiments. For performance evaluation, a consistent 10-fold cross-validation policy is applied for all experiments by using one tenth data as a testing set, and the rest of data as a training set.

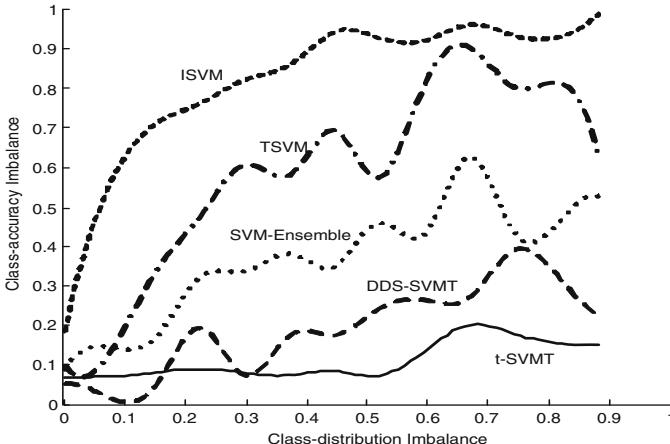
5.1 Robustness Tests on Face Membership Authentication

To evaluate the algorithm's robustness to class-imbalance, we study the face membership authentication (FMA) problem in [4]. The membership authentication problem is to distinguish the membership class from the non-membership class based on the personal facial images stored in a database. An FMA problem is a typical class-imbalance problem, since the membership group is generally much smaller than the nonmembership group. The dataset from [4] includes images collected from 270 individuals (5 images for each).

```

Function: t-SVMTtraining( $P, x$ )
 $x$ ; /* a test instance */
 $\mathbf{K}$ ; /* SVM Kernel for constructing SVM Tree*/
 $T$ ; /* output t-SVMT of  $x$  */
begin
1  $V_t = \emptyset$ ; /* initialize transductive particle set */
2 for each  $Z_k \in P$  {
3   if  $Z_k$  is close to  $x$ 
4      $V_t = V_t \cup [Z_k, \rho_p]$ ; }
5  $T = \emptyset$ ; /* initialize t-SVMT as a root node */
6 for each  $Z_k \in V_t$  {
7   if  $Z_k$  is one-class
8      $M_k = \text{Train\_SVMone}(Z_k, \mathbf{K})$ ; /* one-class SVM*/
9   else
10     $M_j = \text{Train\_SVM}(Z_k, \mathbf{K})$ ; }
11  $T = T \cup [Z_k, M_k, \rho_k]$ ; /* Add a tree node at level  $\rho_j$  */
end

```

Algorithm 2. Transductive Spanning SVM Classification Tree**Fig. 2.** The class-accuracy imbalance versus class-distribution imbalance. t-SVMT is compared with inductive SVM (ISVM), transductive SVM (TSVM), and SVM Ensemble, and DDS-SVMT.

To see how the class-imbalance influence the imbalance of the class accuracy, we gradually decrease the size of the membership class from 135 to 10. Fig. 2 shows the comparison of 5 methods on the class-distribution imbalance versus class-accuracy imbalance, where the imbalance value is defined as $1 - \frac{\min(\text{class1}, \text{class2})}{\max(\text{class1}, \text{class2})}$. As can be seen, as the imbalance of class-distribution increases from $0 (1 - \frac{135}{270-135})$ to $0.9615 (1 - \frac{10}{270-10})$, the class-accuracy imbalance of inductive SVM shows an immediate dramatic increase. TSVM does suppress the class-accuracy imbalance a little, but for most cases the class-accuracy imbalance is above 0.5. SVM ensemble has made the class-accuracy imbalance below

0.5 in most cases, but large fluctuations exist at the later part of the curve. The promising property of DDS-SVMT is that it shows good robustness to class-distribution imbalance, even for class-distribution imbalance greater than 0.6. However the active fluctuation shows the potential instability. The proposed t-SVMT shows the best robustness to class-distribution imbalance such that it keeps a smooth curve with the class-accuracy imbalance kept below 0.2, even for the most imbalanced case when the class-distribution imbalance is at the maximum of 0.962. On the other hand, by comparing the face membership authentication performance, the proposed t-SVMT is shown to clearly outperform single SVMs and SVM ensemble and to be very competitive with respect to the DDS-SVMT in terms of the 10-fold cross validation output. This indicates that the proposed t-SVMT has an outstanding discrimination, but an even better class balancing capability than the previous DDS-SVMT.

5.2 Over-Fitting Tests on Cancer Diagnosis

To see if the t-SVMT can prevent over-fitting to the training dataset, we compared the proposed t-SVMT with DDS_SVMT on the scale of the tree structures, i.e. the number of nodes in the tree, and the classification performance, i.e. the classification accuracy with 10-fold cross validation. The experiments are done on eight well-known two-class cancer datasets. We check the performance of the algorithms under two conditions: For the first case, the raw gene features are used as input, and for the second case, 100 genes selected by a standard t-test gene selection algorithm are used as input.

Table 1. Average classification accuracies for 8 cancer datasets, based on 10-fold cross validation, values are means of 10 runs. Numbers in boldface indicate the best results.

Cancer dataset	Genes/with selection	Bias ratio class 1/2	DDS_SVMT /Tree size	t-SVMT /Tree size
Lymphoma(1)	7129/100	19/58=0.33	84.4% /15	77.9% /6
Leukemia*	7219/100	11/27=0.41	64.7% /24	78.3% /12
CNS Tumour	7129/100	21/39=0.53	50.0% /34	72.0% /8
Colon Cancer	2000/100	22/40=0.55	71.3% /21	80.7% /10
Ovarian	15154/100	91/162=0.56	97.3% /13	75.0% /4
Breast * Cancer	24482/100	34/44=0.77	52.6% /38	73.7% /4
Lymphoma(2)	6431/100	26/32=0.81	51.7% /27	60.3% /10
Lung * Cancer	12533/100	16/16=1.0	64.4% /15	75.0% /8
				73.8% /7

* Independent validation dataset was used for the accuracy evaluation.

Table 1 lists the details and the experimental results of the datasets. As seen from the table, over-fitting happens for DDS-SVMT as it yields big tree structures that have about 30 nodes for datasets with less than 80 instances, such as CNS Tumour, Breast Cancer, and Lymphoma(2). In these cases, the classification accuracies are lower than 50%. On the contrary, the trees created by t-SVMT are normally 2-3 times smaller than those by DDS-SVMT, and outperforms DDS-SVMT in classification accuracy on 12 out of 16 case studies. This indicates that t-SVMT is capable of preventing over-fitting by reducing the size of SVMT, whilst maintains a superior classification accuracy on datasets with significant class imbalance.

6 Discussions and Conclusions

In this paper, we introduced a new type of SVM classification tree that performs effective personalized transductive learning for new test instances, which implements a new type of SVM aggregating intelligence [3] for transductive learning.

The proposed t-SVMT, from the viewpoint of personalized transductive learning, is different from previous PTL methods such as k NN and NFI, as it uses SVM particle based knowledgeable neighborhood instead of the simple neighborhood defined by distance metric. On the other hand, from the viewpoint of SVM aggregating intelligence [3], t-SVMT follows the same route of recursive data partitioning plus aggregating as the previous SVM tree methods. However, t-SVMT is also different from all previous SVMT aggregating methods since it does not consider all the knowledge of the overall dataset, but only the regional knowledge (i.e. knowledge related to the test instance), therefore the proposed t-SVMT solves successfully the over-fitting problem of previous SVMTs.

References

1. Vapnik, V.N.: The Nature of Statistical Learning Theory, 2nd edn., pp. 237–240, 263–265, 291–299. Springer, Berlin (1999)
2. Song, Q., Kasabov, N.: NFI: A Neuro-Fuzzy Inference Method for Transductive Resoning. IEEE Trans. on Fuzzy Systems 13(6), 799–808
3. Pang, S.: SVM Aggregation: SVM, SVM Ensemble, SVM Classification Tree. IEEE SMC eNewsletter (December 2005), <http://www.ieeesmc.org/Newsletter/Dec2005/R11Pang.php>
4. Pang, S., Kim, D., Bang, S.Y.: Face Membership Authentication Using SVM Classification Tree Generated by Membership-based LLE Data Partition. IEEE Trans. on Neural Network 16(2), 436–446 (2005)
5. Schölkopf, J.C., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Technical report, Microsoft Research, MSR-TR-99-87 (1999)
6. Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. In: Procs of the Sixteenth International Conference on Machine Learning, pp. 200–209 (1999)
7. Pang, S., Kim, D., Bang, S.Y.: Membership authentication in the dynamic group by face classification using SVM ensemble. Pattern Recognition Letters 24, 215–225 (2003)

Improving Text Classification Performance with Incremental Background Knowledge

Catarina Silva^{1,2} and Bernardete Ribeiro²

¹ School of Technology and Management,
Polytechnic Institute of Leiria, Portugal

² Dep. Informatics Eng., Center Informatics and Systems,
Univ. of Coimbra, Portugal
catarina@dei.uc.pt, bribeiro@dei.uc.pt

Abstract. Text classification is generally the process of extracting interesting and non-trivial information and knowledge from text. One of the main problems with text classification systems is the lack of labeled data, as well as the cost of labeling unlabeled data. Thus, there is a growing interest in exploring the use of unlabeled data as a way to improve classification performance in text classification. The ready availability of this kind of data in most applications makes it an appealing source of information.

In this work we propose an Incremental Background Knowledge (IBK) technique to introduce unlabeled data into the training set by expanding it using initial classifiers to deliver oracle decisions. The defined incremental SVM margin-based method was tested in the Reuters-21578 benchmark showing promising results.

1 Introduction

Applications of text mining are ubiquitous, since almost 80% of the information available is stored as text. Thus, there is an effective interest in researching and developing applications that better help people handling text-based information. On the other hand, the wealth of text in digital form has made the organization of that information into a complex and vitally important task.

Most text categorization methods, e.g., K-Nearest Neighbor, Naïve Bayes, Neural Nets and Support Vector Machines, have their performance greatly defined by the training set available. To achieve the best classification performance with a machine learning technique, there has to be enough labeled data. However, these data are costly and sometimes difficult to gather. This is one key difficulty with current text categorization algorithms, since they require manual labeling of more documents than a typical user can tolerate [1].

Labeling data is expensive but, in most text categorization tasks, unlabeled data are often inexpensive, abundant and readily available. Therefore, to achieve the purpose of using relatively small training sets, the information that can be extracted from the testing set, or even unlabeled examples, is being investigated as a way to improve classification performance [2,3]. Seeger in [4] presents a report on learning with unlabeled data that compares several approaches.

In general, unlabeled examples are much less expensive and easier to gather than labeled ones. This is particularly true for text classification tasks involving online data sources, such as web pages, email and news stories, where large amounts of text are readily available. Collecting this text can frequently be done automatically, so it is feasible to collect a large set of unlabeled examples. If unlabeled examples can be integrated into supervised learning, then building text classification systems will be significantly faster, less expensive and more effective.

There is a catch however, because, at first glance, it might seem that nothing is to be gained from unlabeled data, since an unlabeled document does not contain the most important piece of information - its classification.

Consider the following example to give some insight of how unlabeled data can be useful. Suppose we are interested in recognizing web pages about conferences. We are given just a few conferences and non-conferences web pages, along with a large number of pages that are unlabeled. By looking at just the labeled data, we determine that pages containing the word *paper* tend to be about conferences. If we use this fact to estimate the classification of the many unlabeled web pages, we might find that the word *deadline* occurs frequently in the documents that are classified in the positive class. This co-occurrence of the words *paper* and *deadline* over the large set of unlabeled training data can provide useful information to construct a more accurate classifier that considers both *paper* and *deadline* as indicators of positive examples.

In this work we propose an Incremental Background Knowledge (IBK) technique that uses the Support Vector Machine (SVM) classification margin to determine unlabeled examples classification and strengthen the training set. The IBK is an improvement of a Basic Background Knowledge (BBK) approach already proposed by the authors in [5]. The new incremental technique provides a more stable convergence to an improved performance without using any new supervisor knowledge.

The rest of the paper is organized as follows. Section 2 addresses several text classification issues, setting guidelines for problem formulation, including the application of Support Vector Machines (SVMs) to text classification tasks.

Section 3 presents the proposed Incremental Background Knowledge (IBK) approach comparing with the previous technique, followed by experimental setup and results in Sections 4 and 5 respectively. Finally, Section 6 presents some conclusions and future work.

2 Text Classification

The goal of text classification is the automatic assignment of documents to a fixed number of semantic categories. Each document can be in multiple, exactly one, or no category at all. Using machine learning, the objective is to learn classifiers from examples, which assign categories automatically. This is usually considered a supervised learning problem. To facilitate effective and efficient learning, each category is treated as a separate binary classification problem. Each of such problems answers the question of whether or not a document should be assigned to a particular category [6].

Documents, which typically are strings of characters, have to be transformed into a suitable representation both for the learning algorithm and the classification task. The most common representation is known as the *Bag of Words* and represents a document by the words occurring in it. Usually the irrelevant words are filtered using a stopword list and the word ordering is not deemed relevant for most applications. Information retrieval investigation proposes that instead of words, the units of representation could be word stems. A word stem is derived from the occurrence form of a word by removing case and inflection information. For example "viewer", "viewing", and "preview" are all mapped to the same stem "view".

This leads to an attribute-value representation of text. Each distinct word w_i corresponds to a feature $TF(w_i, x)$, representing the number of times word w_i occurs in the document x . Refining this basic representation, it has been shown that scaling the dimensions of the feature vector with their inverse document frequency $IDF(w_i)$ leads to an improved performance. $IDF(w_i)$ (II) can be calculated from the document frequency $DF(w_i)$, which is the number of documents the word w_i occurs in.

$$IDF(w_i) = \log \left(\frac{D}{DF(w_i)} \right) \quad (1)$$

Here, D is the total number of documents. The inverse document frequency of a word is low if it occurs in many documents and is highest if the word occurs in only one. To disregard different document lengths, each document feature vector \mathbf{x} is normalized to unit length [7].

2.1 SVM Text Classification

Support Vector Machines (SVMs) are a learning method introduced by Vapnik [8] based on his Statistical Learning Theory and Structural Risk Minimization Principle. When using SVMs for classification, the basic idea is to find the optimal separating hyperplane between the positive and negative examples. The optimal hyperplane is defined as the one giving the maximum margin between the training examples that are closest to it. Support vectors are the examples that lie closest to the separating hyperplane. Once this hyperplane is found, new examples can be classified simply by determining on which side of the hyperplane they are.

Although text categorization is a multi-class, multi-label problem, it can be broken into a number of binary class problems without loss of generality. This means that instead of classifying each document into all available categories, for each pair $\{\text{document}, \text{category}\}$ we have a two class problem: the document either belongs or does not to the category. Although there are several linear classifiers that can separate both classes, only one, the Optimal Separating Hyperplane, maximizes the margin, i.e., the distance to the nearest data point of each class, thus presenting better generalization potential.

The output of a linear SVM is $u = \mathbf{w} \times \mathbf{x} - b$, where \mathbf{w} is the normal weight vector to the hyperplane and \mathbf{x} is the input vector. Maximizing the margin can be seen as an optimization problem:

$$\begin{aligned} & \text{minimize} \quad \frac{1}{2} \|\mathbf{w}\|^2, \\ & \text{subjected to } y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \forall i, \end{aligned} \quad (2)$$

where \mathbf{x} is the training example and y_i is the correct output for the i th training example. Intuitively the classifier with the largest margin will give low expected risk, and hence better generalization.

3 Incremental Background Knowledge

Some authors [9] refer to unlabeled data as background knowledge, defining it as any unlabeled collection of text from any source that is related to the classification task. Joachims presents in [6] a study on transductive SVMs (TSVMs) introduced by Vapnik [8]. TSVMs make use of the testing set and extend inductive SVMs, finding an optimal separating hyperplane not only of the training examples, but also of the testing examples [10].

The Incremental Background Knowledge (IBK) technique we now propose is in fact a development of a Basic Background Knowledge (BBK) approach already proposed by the authors in [5]. We will start by generally describing the basic strategy that will serve as base comparison, and then the incremental approach that constitutes the main contribution of this work (more details on BBK can be found in [5]).

In the BBK, first an inductive SVM classifier (see Section 2.1) is inferred from the training set, and then it is applied to the unlabeled examples. The BBK approach incorporates, in the training set, new examples classified by the SVM with larger margin, which can be assumed as the ones where the SVM classifier presents more confidence. Fig. 1 illustrates an example where four unlabeled examples (black dots) are classified with small and large margins.

Formally, the BBK approach proceeds by incorporating unlabeled examples (only the features, not the classification) from the unlabeled/testing set directly

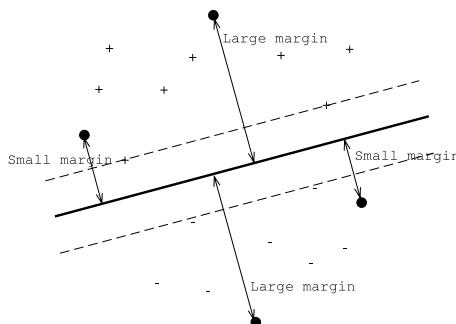


Fig. 1. Unlabeled examples (black dots) with small and large margins

into the training set as classified by the baseline inductive SVM, i.e., an example (\mathbf{x}_i, y_i) will be chosen if Equation (3) holds:

$$(\mathbf{x}_i, y_i) : \rho(\mathbf{x}_i, y_i) = \frac{2}{\|\mathbf{w}\|} > \Delta, \quad (3)$$

Δ was heuristically defined. Notice that Δ is intrinsically related to the margin, i.e. when Δ is decreased, in fact we are decreasing the classification margin of accepted unlabeled examples and thus accepting examples classified with less confidence. This level of confidence should depend on the capabilities of the base classifier, or in other words, the better the base classifier the lower we can set the threshold on Δ (and thus on the margin) to introduce newly classified unlabeled examples into the training set.

In the IBK approach we now suggest, a structural change is proposed to deal with the weaker point of the BBK technique, i.e. the definition of Δ . We proposed the iterative procedure illustrated in Fig. 2. As can be gleaned from this figure, the training set is incrementally constructed by iteratively decreasing the value of Δ , i.e. reducing the confidence threshold for an unlabeled example to be added as classified by the SVM. This approach rational is that as Δ is decreased, the classifiers are also getting better due to the additional information in the training set, thus justifying lowering the confidence threshold. Algorithm 1 below more formally defines the IBK procedure.

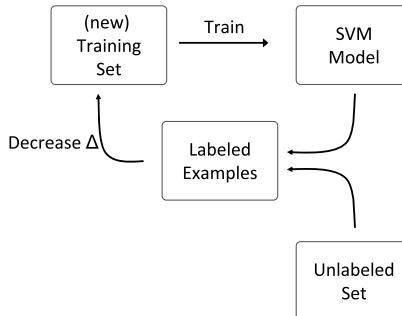


Fig. 2. Proposed approach: Incremental Background Knowledge

Algorithm 1. Incremental Background Knowledge Algorithm

```

Current training set ← Initial dataset
 $\Delta \leftarrow$  initial  $\Delta$  value
WHILE not all unlabeled examples added
  Infer an SVM classifier with current training set
  Classify unlabeled examples with the classifier
  Select the newly classified examples with margin larger than  $\Delta$ 
  Add the selected examples to the current training set
  Decrease  $\Delta$ 
ENDWHILE
  
```

4 Experimental Setup

4.1 Reuters-21578 Benchmark

The widely accepted Reuters-21578 benchmark was used in the experiments. It is a financial corpus with news articles documents averaging 200 words each. Reuters-21578 is publicly available at <http://kdd.ics.uci.edu/databases/reuters-21578/reuters21578.html>. In this corpus 21,578 documents are classified in 118 categories.

Reuters is a very heterogeneous corpus, since the number of documents assigned to each category is very variable. There are documents not assigned to any of the categories and documents assigned to more than 10 categories. On the other hand, the number of documents assigned to each category is also not constant. There are categories with only one assigned document and others with thousands of assigned documents. The ModApte split was used, using 75% of the articles (9603 items) for training and 25% (3299 items) for testing. Table 1 presents the 10 most frequent categories and the number of positive training and testing examples. These 10 categories are widely accepted as a benchmark, since 75% of the documents belong to at least one of them.

Table 1. Number of positive training and testing documents for the Reuters-21578 most frequent categories

Category	Train	Test
Earn	2715	1044
Acquisitions	1547	680
Money-fx	496	161
Grain	395	138
Crude	358	176
Trade	346	113
Interest	313	121
Ship	186	89
Wheat	194	66
Corn	164	52

4.2 Performance Metrics

In order to evaluate a binary decision task we first define a contingency matrix representing the possible outcomes of the classification, as shown in Table 2.

Several measures have been defined based on this contingency table, such as, error rate ($\frac{b+c}{a+b+c+d}$), recall ($\frac{a}{a+c}$), and precision ($\frac{a}{a+b}$), as well as combined measures, such as, the van Rijsbergen F_β measure [11], which combines recall and precision in a single score, $F_\beta = \frac{(\beta^2+1)P \times R}{\beta^2 P + R}$. The latter is one of the best

Table 2. Contingency table for binary classification

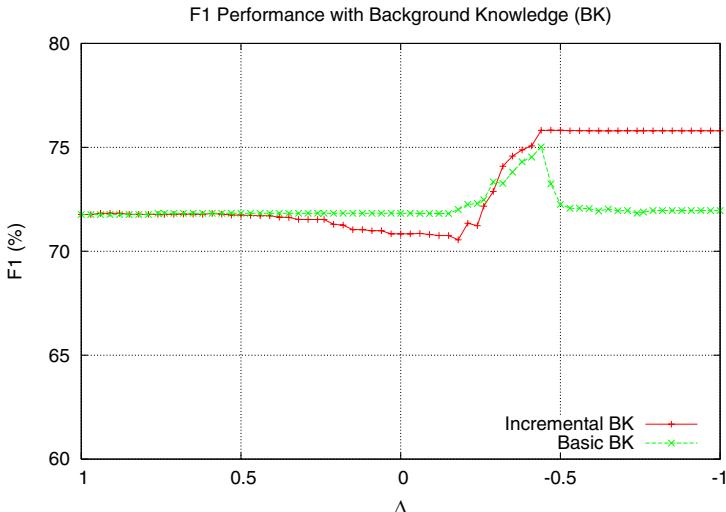
	Class Positive	Class Negative
Assigned Positive	a (True Positives)	b (False Positives)
Assigned Negative	c (False Negatives)	d (True Negatives)

suites measures for text classification used with $\beta = 1$, i.e. F_1 , and thus the results reported in this paper are macro-averaged F_1 values.

5 Experimental Results

Experiments were carried out varying the values of Δ starting with no inclusion of new unlabeled examples until practically all available examples were added. In the BBK approach, for each value of Δ a new training set was constructed, learned and tested, while for IBK the training set in each iteration, corresponding to a value of Δ , was used as baseline for the next iteration, where it was again incremented (see Fig. 2).

Fig. 3 shows the F_1 performance for both approaches and for the several values of Δ . Notice that the values of Δ in x-axis are decreasing values, reflecting the nature of the IBK technique that starts to add large margin classified examples (with high confidence and large Δ values) and proceeds with decreasing values of margin, confidence and Δ . It is clear from this figure that the IBK generally surpasses BBK. However, the most compelling analysis is in its stability

**Fig. 3.** F_1 performance for IBK and BBK for different Δ values

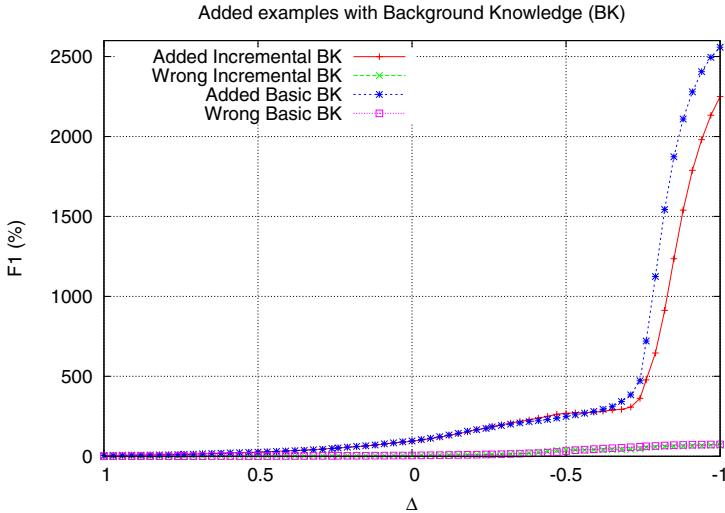


Fig. 4. Total added examples and wrongly added examples for IBK and BBK for different Δ values

to different values of confidence. While BBK presents sensitivity when the confidence drops below an acceptable threshold, IBK remains fairly insensitive and stable, even when all examples are added. It is possible to assert the number of added examples and of these how many are wrongly classified in the graphic in Fig. 4. It is interesting to notice that although more examples are added in the BBK, the number of examples introduced with the wrong classification in the training sets are fairly low and equivalent, despite the difference in classification performance.

6 Conclusions and Future Work

In this work we proposed an Incremental Background Knowledge (IBK) technique that uses the Support Vector Machine (SVM) classification margin to determine unlabeled examples classification and strengthen the training set. The IBK is an improvement of a Basic Background Knowledge (BBK) approach previously developed by the authors.

The main contribution is in the area of semi-supervised learning, by devising a stable and efficient mechanism for automatically incorporating unlabeled examples in the learning task. In fact, the incremental approach does not limit the number of examples introduced and requires no feedback from an oracle. Results shown that the improvement obtained in an optimal point of operation is at least maintained even in severe circumstances of very low level of confidence in the added unlabeled examples.

Future work is expected in further validating the strategy in different applications, namely multiclass applications.

References

1. Schohn, G., Cohn, D.: Less is more: Active Learning with Support Vector Machines. In: International Conference on Machine Learning, pp. 839–846 (2000)
2. Hong, J., Cho, S.-B.: Incremental Support Vector Machine for Unlabeled Data Classification. In: International Conference on Neural Information Processing (ICONIP), pp. 1403–1407 (2002)
3. Liu, B., Dai, Y., Li, X., Lee, W., Yu, P.: Building Text Classifiers Using Positive and Unlabeled Examples. In: International Conference on Data Mining, pp. 179–188 (2003)
4. Seeger, M.: Learning with Labeled and Unlabeled Data, Technical Report, Institute for Adaptive and Neural Computation, University of Edinburgh (2001)
5. Silva, C., Ribeiro, B.: On Text-based Mining with Active Learning and Background Knowledge using SVM. Journal of Soft Computing - A Fusion of Foundations, Methodologies and Applications 11(6), 519–530 (2007)
6. Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. In: International Conference on Machine Learning, pp. 200–209 (1999)
7. Sebastiani, F.: A Tutorial on Automated Text categorisation. In: Amandi, A., Zunino, A. (eds.) Proceedings of ASAII 1999, 1st Argentinian Symposium on Artificial Intelligence, Buenos Aires, AR, pp. 7–35 (1999)
8. Vapnik, V.: The Nature of Statistical Learning Theory, 2nd edn. Springer, Heidelberg (1999)
9. Zelikovitz, S., Hirsh, H.: Using LSI for text classification in the presence of background text. In: Tenth International Conference on Information Knowledge Management, pp. 113–118 (2001)
10. Silva, C., Ribeiro, B.: Labeled and Unlabeled Data in Text Categorization. In: IEEE International Joint Conference on Neural Networks (2004)
11. van Rijsbergen, C.: Information Retrieval, 2nd edn. Butterworths, London (1979)

Empirical Study of the Universum SVM Learning for High-Dimensional Data

Vladimir Cherkassky and Wuyang Dai

Department of Electrical and Computer Engineering,
University of Minnesota, Minneapolis MN 55455 USA
`cherk001@umn.edu, daixx048@umn.edu`

Abstract. Many applications of machine learning involve sparse high-dimensional data, where the number of input features is (much) larger than the number of data samples, $d \gg n$. Predictive modeling of such data is very ill-posed and prone to overfitting. Several recent studies for modeling high-dimensional data employ new learning methodology called Learning through Contradictions or Universum Learning due to Vapnik (1998,2006). This method incorporates a priori knowledge about application data, in the form of additional Universum samples, into the learning process. This paper investigates generalization properties of the Universum-SVM and how they are related to characteristics of the data. We describe practical conditions for evaluating the effectiveness of Random Averaging Universum.

1 Introduction and Background

Sparse high-dimensional data is common in modern machine learning applications. In micro-array data analysis, technologies have been designed to measure the gene expression levels of tens of thousands of genes in a single experiment. However, the sample size in each data set is typically small ranging from tens to low hundreds due to the high cost of measurements. Similarly, in brain imaging studies using magnetic resonance imaging (MRI) and in image recognition studies, the dimensionality d of the data vector is much larger than the sample size n . Such sparse high-dimensional training data sets represent new challenges for classification methods.

Most approaches to learning with high-dimensional data focus on improvements to existing *inductive methods* (Cherkassky and Mulier 2007, Schölkopf and Smola 2002) that try to incorporate a priori knowledge about the good models. Another approach to handling ill-posed high-dimensional classification problems adopts new *non-standard learning formulations* that incorporate a priori knowledge about application data and/or the goal of learning directly into the problem formulation (Cherkassky and Mulier, 2007). Such non-standard learning settings reflect properties of real-life applications, and can result in improved generalization, relative to standard inductive learning. However, these new methodologies

are more complex, and their relative advantages and limitations are still poorly understood.

The idea of ‘inference through contradiction’ was introduced by Vapnik (1998) in order to incorporate a priori knowledge into the learning process. This knowledge is introduced in the form of additional unlabeled data samples (called virtual examples or the *Universum*), that are used along with labeled training samples, to perform inductive inference. Examples from the Universum are not real training samples, however they reflect a priori knowledge about application domain. For example, if the goal of learning is to discriminate between handwritten digit 5 and 8, one can introduce additional ‘knowledge’ in the form of other handwritten digits 0, 1, 2, 3, 4, 6, 7, 9. These examples from the Universum contain certain information about handwritten digits, but they can not be assigned to any of the two classes (5 or 8).

Next we briefly review optimization formulation for the Universum SVM classifier (Vapnik, 2006). Let us consider inductive setting (for binary classification), where we have labeled training data (\mathbf{x}_i, y_i) , $(i = 1, \dots, n)$, and a set of unlabeled examples from the Universum (\mathbf{x}_j^*) , $(j = 1, \dots, m)$. The Universum contains data that belongs to the same application domain as training data, but these samples are *known not to belong* to either class. These Universum samples are incorporated into inductive learning as explained next. Let us assume that labeled training data is linearly separable using large margin hyperplanes $f(\mathbf{x}, \omega) = (\mathbf{w} \cdot \mathbf{x}) + b$. Then the Universum samples can either fall *inside* the margin or *outside* the margin borders (see Fig. 1). Note that we should favor hyperplane models where the Universum samples lie inside the margin, because these samples do not belong to either class. Such Universum samples (inside the margin) are called *contradictions*, because they have non-zero slack variables for either class label. So the Universum learning implements a trade-off between explaining training samples (using large-margin hyperplanes) and maximizing the number of contradictions (on the Universum).

The quadratic optimization formulation for implementing SVM-style inference through contradictions is shown next following (Vapnik, 2006). For labeled training data, we use standard SVM soft-margin loss with slack variables ξ_i . For the Universum samples (\mathbf{x}_j^*) , we need to penalize the real-valued outputs of our classifier that are ‘large’ (far away than zero). So we adopt ϵ -insensitive

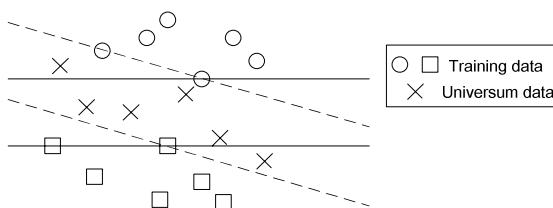


Fig. 1. Two large-margin separating hyperplanes explain training data equally well, but have different number of contradictions on the Universum. The model with a larger number of contradictions should be favored.

loss (as in standard support vector regression). Let ξ_j^* denote slack variables for Universum samples. Then the Universum SVM formulation can be stated as:

$$\begin{aligned} \text{minimize } & R(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C \sum_{i=1}^n \xi_i + C^* \sum_{j=1}^m \xi_j^* \text{ where } C, C^* \geq 0 \\ \text{subject to constraints } & \\ & y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, n \text{ (for labeled data)} \\ & |(\mathbf{w} \cdot \mathbf{x}_j) + b| \leq \varepsilon + \xi_j^*, \quad \xi_j^* \geq 0, \quad j = 1, \dots, m \text{ (for the Universum)} \end{aligned} \quad (1)$$

Parameters C and C^* control the trade-off between minimization of errors and maximizing the number of contradictions. Selecting ‘good’ values for these parameters is a part of model selection (usually performed via resampling). When $C^* = 0$, this U-SVM formulation is reduced to standard soft-margin SVM.

Solution to the above optimization problem defines the large margin hyperplane $f(\mathbf{x}, \omega^*) = (\mathbf{x} \cdot \mathbf{w}^*) + b^*$ that incorporates a priori knowledge (i.e. Universum samples) into the final SVM model. The dual formulation for inductive SVM in the Universum environment, and its nonlinear kernelized version can be readily obtained using standard SVM techniques (Vapnik, 2006). The above quadratic optimization problem is convex due to convexity of constraints for labeled data and for the Universum. Efficient computational algorithms for solving this optimization problem involve modifications of standard SVM software (Weston et al, 2006). Universum SVM software is available at <http://www.kyb.tuebingen.mpg.de/bs/people/fabee/universvm.html>

Successful application of U-SVM depends on implementation of model selection and selection of Universum data. Note that model selection becomes more difficult as kernelized U-SVM has 4 tunable parameters: C , C^* , kernel and ε . In addition, we need to specify the number of Universum samples. In contrast, standard SVM has only two tuning parameters. So in practise, standard SVM may yield better performance than U-SVM, simply because it has inherently more robust model selection.

Selection of Universum samples is usually application-dependent (Vapnik 2006, Weston et al 2006). However, there is a possibility of generating Universum data directly from labeled training data. This approach is called *random averaging* (RA) and it does not rely on a priori knowledge about application domain. Such RA Universum samples are generated by randomly selecting a pair of positive and negative training samples, and computing their average. This paper investigates practical conditions for the effectiveness of U-SVM using random averaging (RA). As Universum samples are generated directly from labeled training data, we expect to express these conditions via the properties of training data. These properties can be conveniently presented using novel representation of high-dimensional training data via univariate histograms introduced in section 2. Section 3 specifies practical conditions for the effectiveness of RA Universum. Section 4 provides empirical examples illustrating the effectiveness of U-SVM learning. Conclusions are presented in Section 5.

2 Representation of High-Dimensional Data via Univariate Projections

Let us consider binary classification problems with sparse high-dimensional data, where the input dimensionality is larger than training sample size ($d \gg n$). Since n points generate n -dimensional subspace (in the input space), the projections of the data points onto any direction vector in the $d - n$ dimensional subspace are all zeros. Also, the projections of the data points onto any vectors orthogonal to the hyperplane generated by the data are non-zero constants. Ahn and Marron (2005) analyzed asymptotic ($d \gg n$) properties of high-dimensional data for the binary classification setting, under the assumption that input variables are ‘nearly independent’. Their analysis suggests that asymptotically there is a direction vector such that the projections of data samples from each class onto this direction vector collapse onto a single point. This projection vector is called the Maximal Data Piling direction vector.

Various linear classifiers differ in approach for selecting the value of the vector \mathbf{w} , specifying the normal direction of a hyperplane $(\mathbf{x} \cdot \mathbf{w}) + b$. For linear SVM classifiers under sparse high-dimensional settings, most data samples (from one class) lie on the margin border, and their projections onto the SVM hyperplane normal direction vector \mathbf{w} tend to be the same (i.e., they project onto the same point). In real-life applications, analytic assumptions in (Ahn and Marron 2005) do not hold, so the data piling effect can be observed only approximately, in the sense that many data samples lie near the margin borders. Next we illustrate the data piling effect using the WinMac text classification data set (UCI KDD 20 Newsgroups entry). This is a binary classification data set where each sample has 7511 binary features. The data is very sparse, and on average only a small portion ($\sim 7.3\%$) of features are non-zeros. We use 200 samples for training, and 200 independent validation samples for tuning linear SVM model parameter C . Fig. 2(a) shows the histogram of univariate projections of the training data onto the normal direction vector \mathbf{w} of the SVM hyperplane. As expected, training data is well separated and training samples from each class cluster near the margin borders, marked as +1 and -1. Also shown in Fig. 2(b) is the histogram of projections of the Universum samples generated from training data via Random Averaging. As training samples cluster at the margin borders, Universum samples will cluster near linear SVM decision boundary (marked 0 on the horizontal axis). In Fig. 2, the y axis of a histogram indicates the number of samples and the histogram of projections are evaluated, separately for each class, by first calculating the range of projected values (i.e., $\max_value - \min_value$), and then dividing this range into 10 different bins. This procedure is used for other histograms of projections shown later in this paper.

For this data set, U-SVM is not likely to provide an improvement over linear SVM, because optimization formulation (1) tries to force the Universum samples to lie near decision boundary. However, as shown in Fig. 2(b), Universum samples already lie near the optimal hyperplane (of standard SVM model), so no additional improvement due to U-SVM can be expected for this data set.

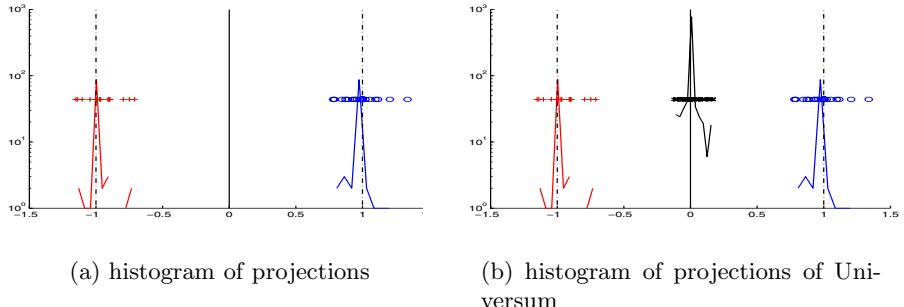


Fig. 2. Histogram of projections of training data onto the normal direction vector \mathbf{w} of the SVM hyperplane

Next we show empirical comparisons between standard linear SVM and U-SVM for the WinMac data set, in order to confirm our intuitive interpretation of Fig. 2. Our comparisons use

- 200 training samples (100 samples per each class);
- 200 independent samples for validation, where validation data set is used for tuning parameters of SVM and U-SVM;
- 1,000 Universum samples generated from training data via random averaging;
- 1,000 independent test samples (used to estimate testing error for each method).

All samples are randomly selected from the WinMac data set, and experiments are repeated 10 times. During model selection, possible values for tuning parameters of U-SVM are given below:

$C \sim [0.01, 0.1, 1, 10, 100, 1000]$, $C^*/C \sim [0.01, 0.03, 0.1, 0.3, 1, 3, 10]$ and $\varepsilon \sim [0, 0.02, 0.05, 0.1, 0.2]$. These parameter values for U-SVM are also used for modeling other data sets presented in this paper.

Performance results in Table II show average training and testing error for each method, where averages are calculated over 10 runs. As expected, Universum SVM shows no improvement over standard linear SVM. Additional information in Table II shows ‘typical’ values of tuning parameters selected by the model selection procedure. Note small values of parameter C^* suggesting that Universum data samples have little effect on the final model. In Table II, the typical value of ε is also shown, but the effectiveness of Universum is mainly determined by the values of C and C^* (or their ratio). So only typical values of parameter C and C^* will be shown later in this paper.

3 Conditions for Effectiveness of Random Averaging Universum

Comparisons for the WinMac data set suggest that it may be possible to judge the effectiveness of RA Universum by analyzing the histograms of projections of

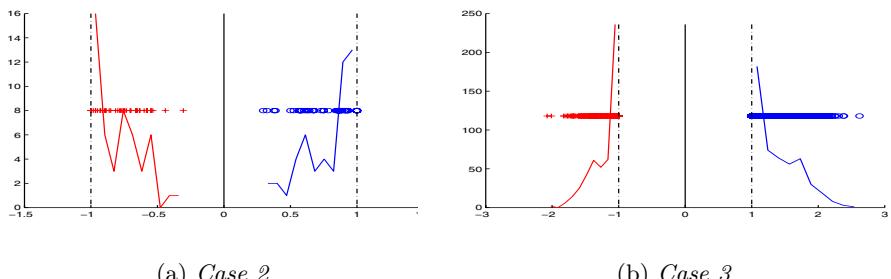
Table 1. Comparison of Linear SVM and U-SVM on WinMac data set. Standard deviation of error rate is shown in parenthesis.

Training/validation set size	200
Average training error rate (SVM)	0
Average training error rate (U-SVM)	0
Average test error rate (SVM)	7.11%(0.92%)
Average test error rate (U-SVM)	7.14%(0.92%)
Ave. Number of Support Vectors (SVM)	195.60
Typical C values	1 or 0.01
Typical C^* values	0.01 or 0.001
Typical ε values	0

training samples onto the normal direction vector \mathbf{w} of standard SVM model. In fact, for sparse high-dimensional training data sets, we can have 3 distinct types of projections:

- *Case 1:* univariate projections of training data onto SVM decision boundary cluster strongly on margin borders (as in Fig. 2).
- *Case 2:* univariate projections of training data onto SVM decision boundary cluster inside margin borders, as shown in Fig. 3(a)
- *Case 3:* univariate projections of training data onto SVM decision boundary cluster outside margin borders, as shown in Fig. 3(b).

From the nature of the U-SVM optimization formulation (1), it can be expected that Universum would not provide any improvement for cases (1) and (2), because Universum samples generated by random averaging are distributed narrowly near SVM decision boundary in the projection space, as shown in Fig. 2(b). However, U-SVM is expected to provide an improvement in case (3), where random averaging would produce Universum samples scattered far away from SVM decision boundary in the projection space, and possibly outside the margin borders of standard SVM.

**Fig. 3.** *Typical histograms:* training data is separable and its projections cluster *inside* or *outside* the margin borders

Note that histograms in Figs. 2, 3 assume that training data is separable. This is generally true for sparse high-dimensional data. For lower-dimensional data, we assume that separability can be achieved using some nonlinear kernel. So the conditions for the effectiveness of RA Universum can be stated as follows:

1. Training data is well-separable (in some optimally chosen kernel space).
2. The fraction of training data samples inside SVM margin borders is small.

The same conditions are also sought by standard SVM classifiers during model selection. That is, optimally selected SVM parameters (kernel and C value) aim at achieving high degree of separation between training samples from two classes. So analysis of univariate histograms for standard SVM model, optimally tuned for a given data set, can be used to ‘predict’ the usefulness of RA Universum.

The univariate histograms of projections of training data for nonlinear kernels are calculated using representation of SVM decision function in the dual space, $f(\mathbf{x}) = \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$. That is, the value of projection of training sample \mathbf{x}_k onto the normal direction of nonlinear SVM decision boundary is expressed as $f(\mathbf{x}_k) = \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k) + b$. The predicted class label for sample \mathbf{x}_k is the sign of $f(\mathbf{x}_k)$. Most existing SVM software packages supply both the label values and real values of the decision function.

In practice, data samples will not always fall *precisely* on the margin borders (denoted as -1 or +1 in the projection space), so the condition for the effectiveness of RA Universum can be quantified via simple separation index:

Separation Index \sim the fraction of training data samples falling in the interval $(-0.99, +0.99)$ in the univariate projection space.

Smaller values of this index, say less than 5-6%, indicate higher separability of the training data, and will generally ensure improved prediction accuracy due to Universum generated via random averaging. This index can be used in practise as ‘rule-of-thumb’ rather than precise necessary condition guaranteeing improved performance of U-SVM vs standard SVM.

4 Empirical Results

This section presents additional empirical comparisons between standard SVM and U-SVM classifiers, using two high-dimensional data sets:

- *Synthetic 1000-dimensional hypercube data set*, where each input is uniformly distributed in $[0,1]$ interval and only 200 out of 1000 dimensions are significant. An output class label is generated as $y = \text{sign}(x_1 + x_2 + \dots + x_{200} - 100)$. For this data set, only linear SVM is used because optimal decision boundary is known to be linear. Training set size is 1,000, validation set size is 1,000, and test set size is 5,000. For U-SVM, 1,000 Universum samples are generated via random averaging from training data.

- *Real-life MNIST handwritten digit data set*, where data samples represent handwritten digit 5 and 8. Each sample is represented as a real-valued vector of size $28 \times 28 = 784$. On average, approximately 22% of the input features are non-zero.

Training set size is 1,000, validation set size is 1,000. For U-SVM, 1,000 Universum samples are generated via random averaging from training data. Separate test set of size 1,866 is used for all experiments.

For each data set, a classifier is applied to training data, its model complexity is optimally tuned using independent validation data set, and then the test error of an optimal model is estimated using test data. The results of such an experiment would depend on random realization of training and validation data. So each experiment is repeated 10 times, using different random realizations, and average error rates are reported for comparison. Linear SVM parameterization is used for synthetic data set, and both linear SVM and nonlinear RBF SVM are used for MNIST data set. Comparison of generalization performance of standard SVM and U-SVM is shown in Table 2, where standard deviation of estimated average test error is indicated in parenthesis.

Table 2. Test error rates for MNIST and synthetic data sets

	SVM	U-SVM
MNIST (RBF Kernel)	1.37%(0.22%)	1.20%(0.19%)
MNIST (Linear Kernel)	4.58%(0.34%)	4.62%(0.37%)
Synthetic data (Linear Kernel)	26.63%(1.54%)	26.89%(1.55%)

These results indicate that U-SVM yields improvement over SVM only for digits data when using RBF kernel. These results can be explained by examining the histograms of projections of training data. Fig. 4 shows the histogram of training data projections onto the normal direction of the RBF SVM decision boundary, suggesting that this data is well-separable. On the other hand, histogram of projections for linear SVM shown in Fig. 5 for both data sets, indicate that training data is not well-separable, so the Universum SVM should not provide any improvement. For MNIST data with RBF SVM, (average) value of separability index is 1.55%, whereas for MNIST data with linear SVM, (average) value of separability index is 15%.

Finally, we investigate the effectiveness of other types of Universum for MNIST data. In this experiment, the training set size is varied as 100, 200 and 1,000;

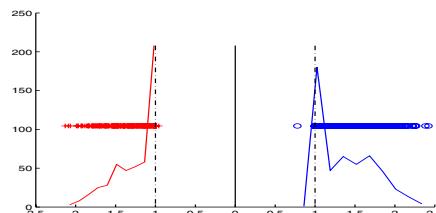


Fig. 4. Histogram of projections of MNIST training data onto normal direction of RBF SVM decision boundary. Training set size~1,000 samples.

and the validation set size is always taken to be the same as training set. For Universum data, 125 samples are randomly selected from each of the digits other than 5 or 8. So the total of 1,000 Universum samples are used. Table 3 presents comparison between standard SVM and Universum SVM using ‘other digits’ as Universum. Table 3 shows that using other digits for Universum always results in improvement over standard SVM. These results (for 1,000 training samples) should be compared with results obtained using random averaging U-SVM reported in Table 2.

In addition, two types of Universum, Random Averaging and Other Digits, are compared for low sample size (100 training samples) in Fig. 6, showing the histograms of projections. As evident from Fig. 6(a), the RA Universum is

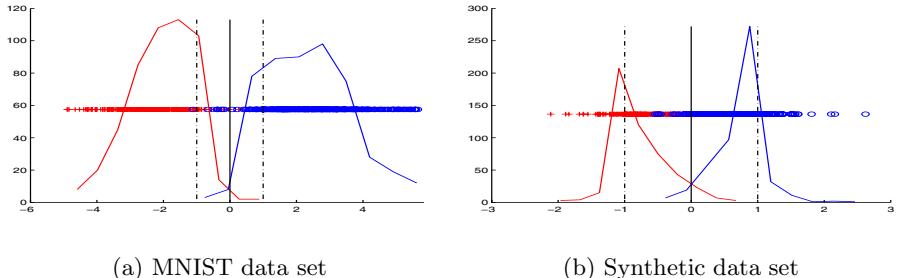


Fig. 5. Histogram of projections onto normal direction of linear SVM hyperplane

Table 3. Test error rates and parameter values of ‘other digits’ Universum SVM

Training set size	100	200	1000
SVM (RBF Kernel)	5.66%(1.89%)	3.69%(0.66%)	1.51%(0.20%)
U-SVM using ‘Other Digits’ Universum	4.86%(2.08%)	3.03%(0.67%)	1.09%(0.26%)
Typical C values selected for SVM	10 or 1 or 0.01	1	10 or 1
Typical C^* values selected for U-SVM	0.1 or 0.01	0.1	3 or 0.3

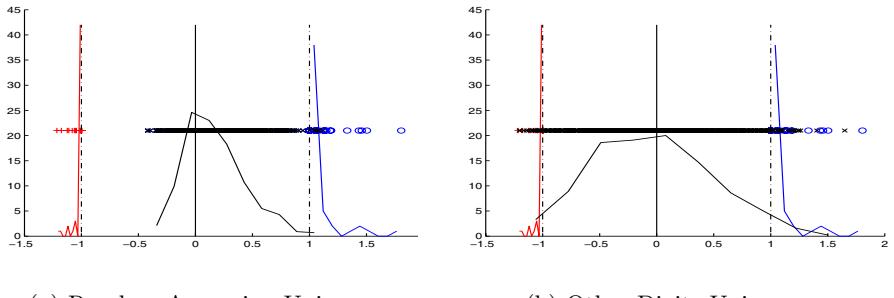


Fig. 6. The histogram of projections of Universum data onto normal direction of RBF SVM decision boundary. Training set size~100 samples.

less effective because its projections are narrowly clustered near SVM decision boundary. On the other hand, projections of the Other Digits Universum are distributed more uniformly between margin borders, suggesting its effectiveness.

Comparison of histograms of projection in Fig. 4 (for 1,000 training samples) and Fig. 6(a) (for 100 samples) shows that the effectiveness of the RA Universum depends on the training sample size.

5 Summary

This paper investigates the effectiveness of the Random-Averaging Universum SVM for high-dimensional data. Our analysis suggests that relative advantages of using U-SVM depend on the properties of training data, such as sample size and noise level. In many situations, using RA Universum SVM does not offer any improvement over standard SVM.

In general, relative performance of learning methods is always affected by the properties of application data at hand. New learning settings, such as Universum-SVM, are inherently more complex than standard SVM, and they have more tuning parameters. So it is important to have simple practical criteria that guarantee potential advantages of using U-SVM, for a given data set. To this end, the paper describes novel representation of high-dimensional training data using projections of this data onto the normal direction of SVM decision boundary. Analysis of the univariate histograms of projected training data, presented in this paper, leads to practical conditions for the effectiveness of RA Universum. Empirical results using several real-life and synthetic data sets illustrate the usefulness of the proposed histogram representation and analysis, for random averaging Universum. The same approach can be used for analyzing effectiveness of other types of Universum, such as ‘other digits’ used for MNIST data set.

Acknowledgments. This work was supported by NSF grant ECCS-0802056, by A. Richard Newton Breakthrough Research Award from Microsoft Corporation, and by BICB grant from the University of Minnesota, Rochester.

References

1. Ahn, J., Marron, J.S.: The direction of maximal data piling in high dimensional space. Technical Report, University of North Carolina at Chapel Hill (2005)
2. Cherkassky, V., Mulier, F.: Learning from Data Concepts: Theory and Methods, 2nd edn. Wiley, NY (2007)
3. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
4. Vapnik, V.N.: Statistical Learning Theory. Wiley, NY (1998)
5. Vapnik, V.N.: Estimation of Dependencies Based on Empirical Data. In: Empirical Inference Science: Afterword of 2006. Springer, Heidelberg (2006)
6. Weston, J., Collobert, R., Sinz, F., Bottou, L.: V.Vapnik, Inference with the Universum. In: Proc. ICML (2006)

Relevance Feedback for Content-Based Image Retrieval Using Support Vector Machines and Feature Selection

Apostolos Marakakis¹, Nikolaos Galatsanos², Aristidis Likas³,
and Andreas Stafylopatis¹

¹ School of Electrical and Computer Engineering,
National Technical University of Athens, 15780 Athens, Greece

² Department of Electrical and Computer Engineering,
University of Patras, 26500 Patras, Greece

³ Department of Computer Science,
University of Ioannina, 45110 Ioannina, Greece

amara@central.ntua.gr, {galatsanos,arly}@cs.uoi.gr, andreas@cs.ntua.gr

Abstract. A relevance feedback (RF) approach for content-based image retrieval (CBIR) is proposed, which is based on Support Vector Machines (SVMs) and uses a feature selection technique to reduce the dimensionality of the image feature space. Specifically, each image is described by a multidimensional vector combining color, texture and shape information. In each RF round, the positive and negative examples provided by the user are used to determine a relatively small number of the most important features for the corresponding classification task, via a feature selection methodology. After the feature selection has been performed, an SVM classifier is trained to distinguish between relevant and irrelevant images according to the preferences of the user, using the restriction of the user examples on the set of selected features. The trained classifier is subsequently used to provide an updated ranking of the database images represented in the space of the selected features. Numerical experiments are presented that demonstrate the merits of the proposed relevance feedback methodology.

Keywords: Content-based image retrieval, relevance feedback, support vector machines, feature selection.

1 Introduction

The target of content-based image retrieval (CBIR)  is to retrieve images relevant to a query of a user, which is expressed by example. In CBIR, an image is described by automatically extracted low-level visual features, such as color, texture and shape. After a user has submitted one or more query images as examples of his/her preferences, a criterion based on this image description is used to rank the images of an image database according to their similarity with the examples of the query and, finally, the most similar are returned to the

user as the retrieval results. Nevertheless, there is an intrinsic difficulty for low-level image features to capture the human perception of image similarity. The reason for this is that the user is usually interested in the semantic content of the images. Unfortunately, the semantic content of an image is very difficult to describe using only low-level image features. This is the well-known semantic gap problem.

Relevance feedback (RF) has been proposed as a technique which can be used in order to bridge the semantic gap. RF is an interactive process between the user and the retrieval system. In each RF round, the user assesses the previously retrieved images as relevant or irrelevant to the initial query and provides this assessment as feedback to the system. This feedback is used, subsequently, by the system so that the ranking criterion is updated and a new set of images is retrieved. In this way, the subjective human perception of image similarity is incorporated to the system and the retrieval results are expected to improve, according to the user's viewpoint, with the RF rounds. With regard to RF approaches proposed in the literature, much work has been done during the last years, e.g. [2], [3], [4], [5], [6]. Among the proposed RF methodologies, the most prominent are those which use classifiers to distinguish between the classes of relevant and irrelevant images, e.g. [2], [3], [4]. In this context, the images assessed by the user as relevant or irrelevant up to the current RF round are used as positive and negative examples, respectively, to train a classifier. This classifier is used, subsequently, to update the database image ranking. It must be mentioned here that, among all the learning models proposed for this classification task, Support Vector Machines (SVMs) [12] constitute the most popular one.

The users of a CBIR system are usually not patient enough to provide the system with a large number of examples in each RF round. On the other hand, the description of the images is, generally, of very high dimensionality. From the above, it becomes obvious that, for RF methodologies which are based on training a classifier using the feedback examples, the available training set of the classifier is, almost always, very small compared to the dimensionality of training patterns. This can deteriorate the classifier performance, leading to poor retrieval results. To alleviate this problem, a methodology for selecting a relatively small number of good features for the classification task, based on the properties of the feedback examples, can be used. In this way, a significant dimensionality reduction can be achieved by removing irrelevant or redundant features. Training the classifier on the resulting lower-dimensional feature space can improve its ability to capture the underlying data distribution, thus leading to better classifier performance. Moreover, by reducing feature space dimensionality and, hence, the complexity of data, a decrease in training and image re-ranking time can be achieved. Many feature selection methods for classification have been proposed in the literature, e.g. [7], [8], [9]. As will be shown below, some of these methods can be incorporated straightforwardly in a CBIR system using RF.

The rest of the paper is organized as follows. The SVM methodology in the context of CBIR using RF is described in Section 2. Section 3 presents the feature selection methods used in this work. In Section 4, the details and results of the

experiments are provided and, finally, in Section 5, conclusions and directions for future research are presented.

2 Using SVMs for RF in CBIR

Consider the binary classification problem $\{(x_i, y_i)\}_{i=1}^N$, where x_i are the labeled patterns and $y_i \in \{-1, +1\}$ the corresponding labels. Based on this training set, we want to train an SVM classifier. The SVM classifier maps the patterns to a new space, called kernel space, using a transformation $x \mapsto \varphi(x)$, in order to get a potentially better representation of them. This new space can be non-linear and of much higher dimension than the initial one. After the mapping, a linear decision boundary is computed in the kernel space. In the context of SVM methodology, the problem of classification is addressed by maximizing the margin, which is defined as the smallest distance, in the kernel space, between the decision boundary and any of the training patterns. This can be achieved by solving the following quadratic programming problem:

$$\max \left[\sum_{i=1}^N a_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N a_i a_j y_i y_j k(x_i, x_j) \right] \quad \text{over } a_i, i = 1, \dots, N \quad (1)$$

$$\text{s.t. } 0 \leq a_i \leq C \quad \text{and} \quad \sum_{i=1}^N a_i y_i = 0 \quad (2)$$

where

$$k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j) \quad (3)$$

is the kernel function and C is a parameter controlling the trade-off between training error and model complexity. The most popular non-linear kernel functions used for SVMs belong to the class of Radial Basis Functions (RBFs). From all RBF functions, the most commonly used is the Gaussian RBF, which is defined by:

$$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \quad (4)$$

After the training of the classifier, the value of the decision function for a new pattern x is computed by:

$$y(x) = \sum_{i=1}^N a_i y_i k(x_i, x) + b \quad (5)$$

where b is a bias parameter the value of which can be easily determined after the solution of the optimization problem (see [12]). The value $|y(x)|$ is proportional to the distance of the input pattern x from the decision boundary. Thus, the value $y(x)$ can be regarded as a measure of confidence about the class of x , with large positive values (small negative values) strongly indicating that x belongs to the class denoted by $+1$ (-1). On the contrary, values of $y(x)$ around zero provide little information about the class of x .

In the framework of CBIR with RF, in each round of RF we have to solve a classification problem as the one described above, where a number of images, represented as feature vectors, correspond to the feedback examples provided by the user so far, and each image is labeled by -1 or $+1$ corresponding to irrelevant or relevant, respectively. The initial query is considered to be one of the relevant images and is labeled by $+1$. From the above, it is obvious that we can train an SVM classifier based on the feedback examples and use it to distinguish between the classes of relevant and irrelevant images. Each image in the database will be presented to the trained classifier and the value of the decision function (Eq. (5)) will be used as the ranking criterion. The higher the value of the decision function for an image, the more relevant this image is considered by the system.

3 Feature Selection

Assume, again, we have the binary classification problem presented above, each pattern x_i being a d -dimensional vector of features. Feature selection consists in reducing the dimensionality of the patterns, usually before training the classifier, by removing those features which are irrelevant or redundant for distinguishing between the training set categories, while keeping informative and important features. As far as the problem of re-ranking the database images can be considered as a binary classification problem, feature selection techniques can be applied in each RF round.

Specifically, in this work, we propose an RF scheme for CBIR, which uses SVMs for the RF task along with the methodology introduced in [7] for feature selection. The proposed feature selection methodology is described next. Additionally, another popular and promising methodology for feature selection is considered, which can be incorporated in the RF scheme in exactly the same way. This variation will be used for reasons of comparison with the proposed method.

3.1 Recursive Feature Elimination Using SVMs (SVM-RFE)

The feature selection methodology proposed in [7], called SVM Recursive Feature Elimination (SVM-RFE), is based on a recursive elimination of the less important features, based on the results of classification of the training patterns using SVM classifiers. Thus, the learning model used by this methodology for feature selection is the same as that adopted for the task of RF in this work. This results to the benefit of selecting those features which are the most important for the subsequent training of the SVM classifier used for RF.

Specifically, the SVM-RFE methodology is based on linear-kernel SVMs. Considering a linear SVM kernel:

$$k(x_i, x_j) = x_i^T x_j \quad (6)$$

Eq. (5), for the decision function, takes the form:

$$y(x) = w^T x + b \quad (7)$$

with

$$w = \sum_{i=1}^N a_i y_i x_i \quad (8)$$

where the vector w is of the same dimensionality as the training patterns x_i . This form of decision function implies that the higher the value $|w_k|$ or w_k^2 for the k -th coordinate of the vector w , the larger is the influence of this coordinate on the value of the decision function for an unknown pattern x . This notion provides us with a criterion which can be used to rank the image features according to their importance for the classification task.

SVM-RFE is a recursive method. In each repetition, it updates a feature set, S_f , which initially includes all the available features, by eliminating the less important feature of the set. To determine the less important feature, it trains an SVM classifier with a linear kernel, using the training patterns restricted on the features currently included in S_f . After training, the feature with the smaller value w_k^2 is considered the less important one and is eliminated from S_f . This procedure is repeated until a predefined number of features remain in S_f . These are the features selected by the method.

3.2 Minimum Redundancy Maximum Relevance (mRMR)

Another popular feature selection technique, called minimum Redundancy Maximum Relevance (mRMR), is proposed in [8]. This methodology is based on mutual information, which is a measure of relevance between two random variables x, y . The mutual information is defined by:

$$I(x; y) = \int \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy \quad (9)$$

where $p(x)$, $p(y)$ and $p(x, y)$ are the probability density function (pdf) of x , the pdf of y and the joint pdf of x, y , respectively. In the framework of mRMR, the d pattern features and the pattern label are considered to be random variables. Under these assumptions, the task consists in the incremental selection of those features which have large relevance with the training labels and, at the same time, have small redundancy among them. This notion is expressed formally by:

$$\max H(f_j) \quad \text{over} \quad f_j \in X - S_f \quad (10)$$

with

$$H(f_j) = D(f_j) - R(f_j) \quad (11)$$

$$D(f_j) = I(f_j; y) \quad (12)$$

$$R(f_j) = \frac{1}{\text{card}(S_f)} \sum_{f_i \in S_f} I(f_j; f_i) \quad (13)$$

where f_i (or f_j), y denote the random variables corresponding to the features and the label, respectively. In this case, S_f is the set of features which have been

selected until now and X is the set of all features. Initially, S_f is empty. In the first repetition, the feature $f_j \in X$ with the maximum value for $D(f_j)$, i.e. the most relevant with the labels, is selected and inserted in S_f . In each subsequent repetition, a feature f_j from the current set $X - S_f$ is inserted in S_f . This feature is one with a sufficiently large relevance with the labels, $D(f_j)$, and a sufficiently small mean relevance with the already selected features in S_f , $R(f_j)$ (which is used as a measure of redundancy), so as to maximize $H(f_j)$. This procedure is repeated until the set of selected features, S_f , contains the desired number of features.

When the random variables are discrete, the mutual information between them can be computed very easily, as the integrals in Eq. (9) are converted to summations for all the possible values of the random variables. The values of the pattern label are naturally discrete. The values of the pattern features can be easily discretized by computing for each feature the mean (μ) and the standard deviation (σ) of the values it takes for all training examples. Then, a common label is assigned for all values in each one of the intervals $(-\infty, \mu - \sigma]$, $(\mu - \sigma, \mu + \sigma]$ and $(\mu + \sigma, +\infty)$. After discretization, the probabilities needed for the mutual information computation can be determined by simply counting the corresponding instances in the training set.

4 Experiments

In order to assess the performance of the proposed method, an image set containing 3740 images from the image database in [3] is used. These images are manually classified into 17 semantic categories, and this categorization will be the ground truth of the RF simulations.

For each image, color, texture and shape information is extracted. As color features we used a 256-dimensional histogram in HSV color space, with quantization $8 \times 8 \times 4$ for the color coordinates H, S and V, respectively, and a 9-dimensional color moment vector in CIE-Lab color space, containing the first 3 moments (mean, standard deviation and skewness) for each one of the 3 color coordinates L*, a* and b*. As texture features we used the 104-dimensional vector produced by the 3-level tree-structured wavelet transform [10], which includes, for each one of the 4 sub-bands resulted by each signal decomposition, the corresponding mean and standard deviation of the energy, and is based on recursive decomposition of the first 3 sub-bands of lower frequency. Finally, as shape features we used the 80-dimensional edge histogram [11], which is formed by the uniform partitioning of the image into 16 sub-images and the computation, for each one of them, of the frequency of occurrence of 5 types of edges (horizontal, vertical, 45 degrees diagonal, etc.). All the above vectors are merged in a single 449-dimensional one, which is the final representation of the image.

We implemented an RF simulation scheme, using Precision as measure of performance, which is the ratio of relevant images within the top T retrieved images. A retrieved image is considered relevant (irrelevant) if it belongs to the same (different) category as (than) the initial query. In this simulation scheme,

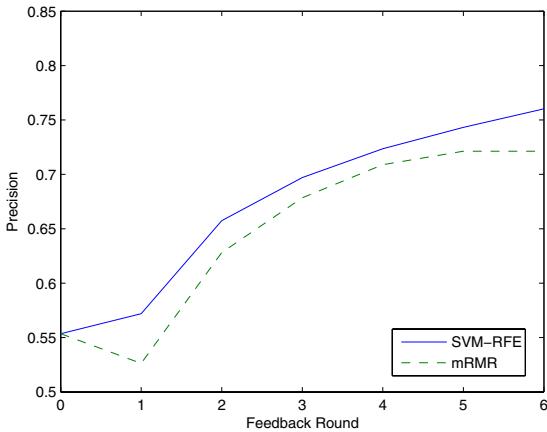


Fig. 1. Average Precision in scope $T = 20$, for $K = 25$ selected features

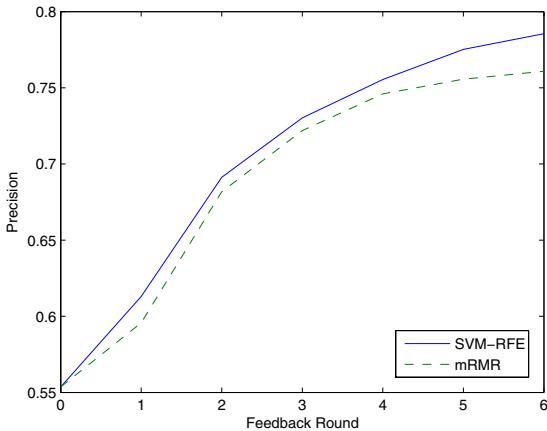


Fig. 2. Average Precision in scope $T = 20$, for $K = 50$ selected features

1000 database images are used once as initial queries. For each initial query, we simulated 6 rounds of RF. In each RF round, at most 3 relevant and 3 irrelevant images are selected randomly from the first 50 images of the ranking. These images are used in combination with the examples provided in the previous RF rounds to select a number, K , of important features and, then, to train a new SVM classifier in the resulting lower-dimensional feature space. Based on this new classifier, the ranking of the database images is updated. For the initial ranking, when no feedback examples have been provided yet and, hence, neither feature selection nor classifier training can be employed, the euclidean distance in the initial feature space is used.

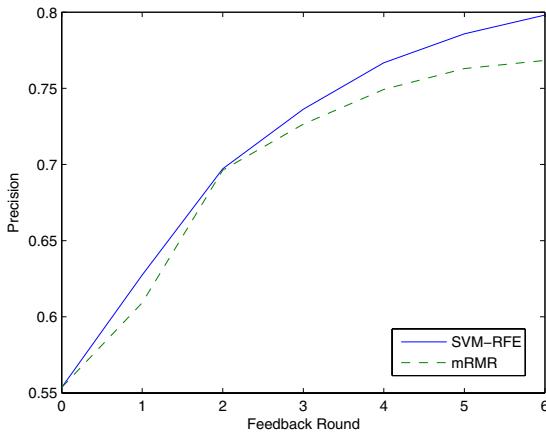


Fig. 3. Average Precision in scope $T = 20$, for $K = 75$ selected features

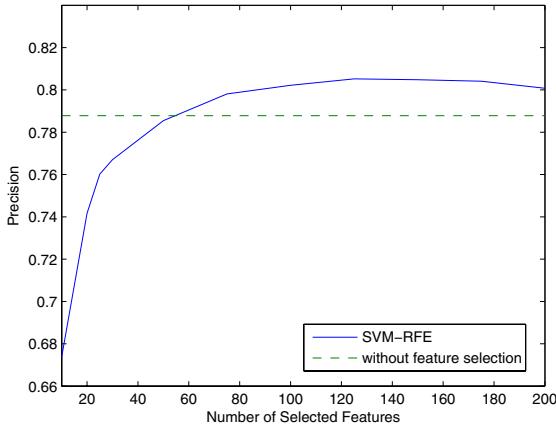


Fig. 4. Average Precision in scope $T = 20$, for the 6th RF round

The first comparison is performed between the two previously described feature selection methodologies, used along with SVMs for the task of CBIR using RF. The number of features to retain is passed as a parameter to the methods. Figures 1, 2 and 3 show the results of the RF simulations, i.e. the average Precision in scope $T = 20$ during different RF rounds, for selected features $K = 25$, $K = 50$ and $K = 75$, respectively. As can be seen, the proposed method constantly outperforms the mRMR variation. Moreover, regarding the computation time required by the two methods, it must be mentioned that the proposed method is much faster. In particular, when $K = 50$, the average time (on a 3GHz PC) consumed by the proposed method per initial query is 2.5 sec for 6 rounds of RF, whereas the mRMR variation needs 15 sec per initial query.

In a second set of experiments, the performance of the proposed method is compared with that obtained when no feature selection is used. Specifically, Figure 4 displays the average Presicion in scope $T = 20$, for the 6th round of RF, when the proposed method is used with different values of K (e.g., 10, 25, 50, 100, 200 etc.), in comparison with that obtained without feature selection. It can be easily seen that, using a relatively small number of features, we can achieve equal or even better performance with respect to that obtained when using the full feature set.

In the experiments, a Gaussian RBF kernel is adopted for the SVM classifiers (except for the case of SVM-RFE method, which assumes linear kernel SVMs). The values of the SVM parameter C and the Gaussian RBF kernel parameter γ are empirically chosen for each experimental setup, so as to obtain the best performance. We used the SVM implementation provided in [14], and the mRMR implementation in [15].

5 Conclusions – Future Work

A new relevance feedback approach for CBIR is presented in this paper. This approach uses SVM classifiers to distinguish between the classes of relevant and irrelevant images, along with an SVM-based feature selection technique to reduce the feature space dimensionality according to the feedback examples. The experimental results demonstrate the superiority of the proposed method compared to an approach based on a very popular feature selection methodology. Furthermore, as indicated by our experiments, even with a very large reduction of the features, a performance equivalent or even better compared to that obtained for the full feature set can be achieved.

In the future, we aim to use more sophisticated image features to represent the image content. Furthermore, we aim to apply techniques for automatic determination of the most appropriate number of features for each round of RF. Finally, we would like to test the scalability of the proposed method using even larger image databases.

Acknowledgement

This work was supported by Public Funds under the PENED 2003 Project co-funded by the European Social Fund (80%) and National Resources (20%) from the Hellenic Ministry of Development - General Secretariat for Research and Technology.

References

1. Datta, R., Li, J., Wang, J.Z.: Content-Based Image Retrieval: Approaches and Trends of the New Age. *Multimedia Information Retrieval*, 253–262 (2005)
2. Guo, G.D., Jain, A.K., Ma, W.Y., Zhang, H.J.: Learning Similarity Measure for Natural Image Retrieval with Relevance Feedback. *IEEE Trans. Neural Netw.* 13(4), 811–820 (2002)

3. Jing, F., Li, M., Zhang, H.-J., Zhang, B.: Relevance Feedback in Region-Based Image Retrieval. *IEEE Trans. Circuits Syst. Video Technol.* 14(5), 672–681 (2004)
4. Tong, S., Chang, E.: Support Vector Machine Active Learning for Image Retrieval. *ACM Multimedia*, 107–118 (2001)
5. Hsu, C.T., Li, C.Y.: Relevance Feedback Using Generalized Bayesian Framework with Region-Based Optimization Learning. *IEEE Trans. Image Process.* 14(10), 1617–1631 (2005)
6. Marakakis, A., Galatsanos, N., Likas, A., Stafylopatis, A.: Probabilistic Relevance Feedback Approach for Content-Based Image Retrieval Based on Gaussian Mixture Models. *IET Image Process.* 3(1), 10–25 (2009)
7. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning* 46, 389–422 (2002)
8. Peng, H., Long, F., Ding, C.: Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* 27(8), 1226–1238 (2005)
9. Wang, L.: Feature Selection with Kernel Class Separability. *IEEE Trans. Pattern Anal. Mach. Intell.* 30(9), 1534–1546 (2008)
10. Chang, T., Kuo, C.-C.J.: Texture Analysis and Classification with Tree-Structured Wavelet Transform. *IEEE Trans. Image Process.* 2(4), 429–441 (1993)
11. Won, C.S., Park, D.K., Park, S.-J.: Efficient Use of MPEG-7 Edge Histogram Descriptor. *ETRI Journal* 24(1), 23–30 (2002)
12. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, Heidelberg (2006)
13. Microsoft Research Cambridge Object Recognition Image Database, version 1.0, <http://research.microsoft.com/research/downloads/Details/b94de342-60dc-45d0-830b-9f6eff91b301/Details.aspx>
14. LIBSVM – A Library for Support Vector Machines, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
15. mRMR Feature Selection Site, <http://research.janelia.org/peng/proj/mRMR/>

Understanding the Principles of Recursive Neural Networks: A Generative Approach to Tackle Model Complexity

Alejandro Chinea

Departamento de Física Fundamental, Facultad de Ciencias UNED,
Paseo Senda del Rey nº9, 28040-Madrid, Spain

Abstract. Recursive Neural Networks are non-linear adaptive models that are able to learn deep structured information. However, these models have not yet been broadly accepted. This fact is mainly due to its inherent complexity. In particular, not only for being extremely complex information processing models, but also because of a computational expensive learning phase. The most popular training method for these models is back-propagation through the structure. This algorithm has been revealed not to be the most appropriate for structured processing due to problems of convergence, while more sophisticated training methods enhance the speed of convergence at the expense of increasing significantly the computational cost. In this paper, we firstly perform an analysis of the underlying principles behind these models aimed at understanding their computational power. Secondly, we propose an approximate second order stochastic learning algorithm. The proposed algorithm dynamically adapts the learning rate throughout the training phase of the network without incurring excessively expensive computational effort. The algorithm operates in both on-line and batch modes. Furthermore, the resulting learning scheme is robust against the vanishing gradients problem. The advantages of the proposed algorithm are demonstrated with a real-world application example.

Keywords: Recursive neural networks, structural patterns, machine learning, generative principles.

1 Introduction

Recursive neural networks were introduced last decade [1][2] as promising machine learning models for processing data from structured domains (i.e.: protein topologies, HTML web pages, DNA regulatory networks, parse trees in natural language processing, and image analysis amongst others). These computational models are suited for both classification and regression problems being capable of solving supervised and non-supervised learning tasks. One of the goals behind this approach was to fill the existing gap between symbolic and sub-symbolic processing models. Specifically, to develop computational schemes able to combine numerical and symbolic information in the same model. Moreover, the principal advantage associated to them was their ability to work with patterns of

information of different sizes and topologies (i.e. trees or graphs) as opposed to feature-based approaches where the information pertinent to the problem is encoded using fixed-size vectors. Graphs are more flexible data structures than vectors since they may consist of an arbitrary number of nodes and edges, while vectors are constrained to a predefined length which has to be preserved by all patterns composing the training set. So far, these models have found applications mainly in bio-informatics [3][4] although they have also been applied to image analysis [5] and natural language processing tasks [6] between others.

However, despite the initial interest motivated by these models their development is still in its infancy. This fact can be explained in part because the inherent restrictions associated to the first models where the recursive paradigm was limited to work only with acyclic structures under causality and stationary assumptions, something too restrictive for many real-world problems. Another limiting factor to be considered is that structured domains possess very little mathematical structure. Specifically, basic mathematical operations such as computing the sum or the covariance of two graph objects are not available. Furthermore, it is also important to note that nowadays, early research problems like learning generic mappings between two structured domains (e.g.: IO-isomorphic and non IO-isomorphic structured transductions) still remains as challenging open research problems.

Although some advances have been recently reported regarding not only the recursive processing of cyclic structures [7][8] but the contextual processing of information (i.e. recursive models breaking the causality hypothesis) [9][10], followed by some basic proposals on generating structured outputs [11][12], from a practical point of view, the intrinsic complexity of these models together with a computationally hard learning phase has strongly limited the interest of the research community on this kind of models. It is important to note that learning in structured domains has been traditionally considered a very difficult task. Furthermore, it has been recently pointed out [13] that two important future challenges for these models will rely on the design of efficient learning schemes, and tackling appropriately theoretical problems as learning structural transductions or structure inference as occur in various machine learning areas such as the inference of protein structures or parse trees.

In this paper we present an approximate stochastic second order training algorithm aimed to overcome the complexity of the training phase associated to these models. The main advantage of this training method is that is robust against the vanishing gradients problem. Furthermore, the resulting scheme leads to an algorithm which achieves an optimal trade-off between speed of convergence and the required computational effort. In addition, this paper puts also the emphasis on the analysis of the underlying principles of the computational model associated to recursive neural networks. This analysis will permit us to better understand their computational power. The rest of the paper is organized as follows: In the next section we analyze in detail the principles behind the computational model implemented by recursive neural networks. Section 3, is devoted to show

the background of the approximate second order stochastic algorithm. In section 4, experimental results are provided comparing the proposed algorithm with existing approaches. Finally, concluding remarks are outlined in section 5.

2 Implications of the Recursive Neural Model

2.1 Definitions and Notations

A graph U is a pair (V, E) , where V is the set of nodes and E represents the set of edges. Given a graph U and a vertex $v \in V$, $pa[v]$ is the set of parents of v , while $ch[v]$ represents the set of its children. The in-degree of v is the cardinality of $pa[v]$, while its out-degree is the cardinality of $ch[v]$. Under the recursive model the patterns of information are labeled graphs. Specifically, the graph nodes contain a set of domain variables characterized by a vector of real and categorical variables. Furthermore, each node encodes a fragment of information that is believed to play an important role in the task at hand. The presence of a branch (v, w) between two nodes explicitly models a logical relationship between the fragments of information represented by nodes v and w .

The recursive neural network model is composed of a state transition function f and an output function g (see figure ②). These functions are usually implemented by multi-layer perceptron networks. The standard model is suited to process directed positional acyclic graphs with a super-source node. Furthermore, they implement deterministic IO-isomorphic transductions based on the following recursive state representation:

$$\begin{aligned} a(v) &= f(a(ch[v]), I(v), v, W_f) \\ y(v) &= g(a(v), v, W_g) \end{aligned} \quad (1)$$

In expression ② W_f and W_g represent the synaptic weights (model parameters) of networks f and g respectively. In order to process a graph U the state transition network is unfolded through the structure of the input graph leading to the encoding network. This unfolding procedure is followed in both learning and recall phases of the neural network. The resulting network has the same structure of the input graph, while nodes are replaced by copies of the state transition network and a copy of the output network is inserted at the super-source. Afterwards, a feed-forward computation is carried out on the encoding network. More specifically, at each node v of the graph, the state $a(v)$ is computed by the transition network as a function of the input label $I(v)$ and the state of its children (first equation of expression (1)) with:

$$a(ch[v]) = [a(ch_1[v], a(ch_2[v], \dots, a(ch_o[v])])] \quad (2)$$

In expression ② the index o stands for the maximum out-degree of node v . The base case for the recursion is $a(\text{nil}) = a_0$ which correspond to a frontier state (e.g. if node v lacks of its i -th child). At the super-source (node s) the output

is computed as $y = g(a(s))$. It is important to note that the standard model of recursive neural networks implements transductions consisting on the mapping of input structures U into an output structure Y which is always made of a single node (the output is not structured).

2.2 Generative Principles of Intelligence

Generally speaking, the notion of intelligence cannot be conceived without the existence of two fundamental principles [14]: The maximization of transfer and the recoverability principle. Basically, the first principle states that whenever it is possible an intelligent system builds knowledge by maximizing the transfer of previously acquired knowledge. Specifically, more complex structures are the result of assembling previously learnt or memorized structures (e.g. to transfer actions used in previous situations to handle new situations). However, this mechanism of transfer must be understood as a gestalt procedure (the resulting structures being more than a simply combination of previously learnt structures). In addition, the recoverability principle is associated to the concept of memory and inference. It states that a system displaying intelligence must be able to recover itself from its actions. Specifically, an intelligent system must be able to infer causes from its own current state in order to identify what it failed or succeed something not possible without the existence of memory.

It has been recently shown [15] that the human information processing system follows these two generative principles. Specifically, the perception system organizes the world by using cohesive structures. Furthermore, such a structure is the result of a hierarchically organized information processing system [16][17] that generates structure by correlating the information processed at the different levels of its hierarchy. As a result of this process, world complexity is turned into understanding by finding and assigning structure. This mechanism, amongst many other things, permits us to relate objects of different kinds. In addition,

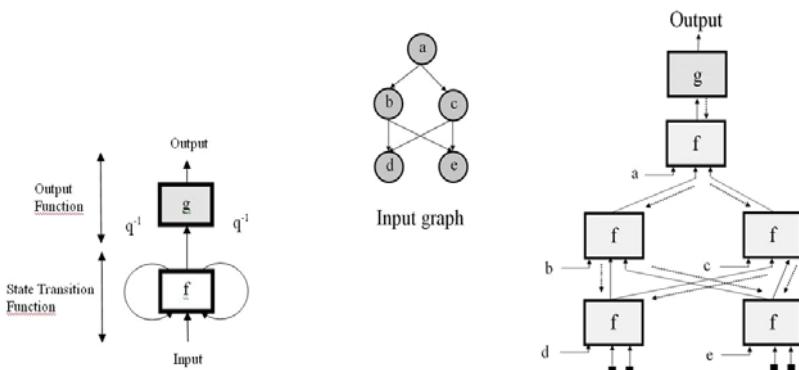


Fig. 1. Block diagram of the Recursive neural network model (left side of the figure). The right side of the picture depicts the unfolding procedure.

the nature of the perceived structures and their relationships is also linked to the context under consideration. For instance, the relationships between objects can be causal in a temporal context, geometrical in pattern recognition problems (e.g. the shapes appearing in an image) or topological in bio-informatics problems (e.g. chemical compounds, protein structures etc).

2.3 Recursive Networks as Generative Models

Under the recursive neural networks framework the perceived structure of a problem is captured and expressed by using graphical models. In particular, the patterns used for the learning and recall phases not only encode the fragments of information (e.g.: information that can be characterized by specific attributes that are quantifiable and/or measurable) which play an important role in the machine learning problem but also the logical relationships between them. The nature of such relations is determined by the application context and attempts to explicitly model the logical correlations between fragments of information. For instance, in a temporal domain the fragments of information are events and the co-occurrence of two or more events is interpreted as an existing or possible correlation between them. Therefore, this information encoding procedure contains more knowledge rather than if such pieces of information were considered in isolation. It is important to note that the notion of information content is strongly connected to the notion of structure. Indeed, the fact of building more complex structures from more basic ones is reflecting the first of the two generative principles related to the concept of intelligence resulting in a gain not only in information representation but in information content.

On the other hand, the computational scheme imposed by the recursive state equation (II) leads to a sequential propagation of information which follows a reversed topological sort of the input graph during the recall phase of the network (hidden states are updated starting from leaves toward the super-source node). In addition, this flow of information is bidirectional during the learning phase. The main consequence of this message passing procedure is that the computational model of recursive networks can be viewed as an inference system that learns the hidden dependencies explicitly encoded within the structural patterns used for the training phase of the network. Furthermore, as stated in [18] recursive neural networks can be viewed as limits, both in distribution and probability, of Bayesian networks with local conditional distributions. Therefore, they implement in a simplified form the notion of recoverability.

3 Reducing Complexity of the Training Phase

3.1 Theoretical Background

The concept of learning in neural networks is associated to the minimization of some error function $E(W)$ by changing network parameters W . In the case of recursive networks the learning phase consists in finding the appropriate

model parameters for implementing the state transition network f and the output network g with regards to the given task and data. Without a loss of generality let us suppose that we express the parameters of function f and g in a vector $W = [w_1, w_2, w_3, \dots, w_m]$. A perturbation of the error function around some point of the model parameters which can be written as follows: $E(W + \Delta W) = E(w_1 + \Delta w_1, w_2 + \Delta w_2, \dots, w_m + \Delta w_m)$. Considering the Taylor expansion of the error function around the perturbation ΔW we obtain:

$$E(W + \Delta W) = E(W) + \sum_{i=1}^m \frac{\partial E(W)}{\partial w_i} \Delta w_i + \frac{1}{2} \sum_{i=1}^m \frac{\partial^2 E(W)}{\partial w_i^2} (\Delta w_i)^2 + \sum_{i < j} \frac{\partial^2 E(W)}{\partial w_i \partial w_j} \Delta w_i \Delta w_j + \frac{1}{6} \sum_{i=1}^m \frac{\partial^3 E(W)}{\partial w_i^3} (\Delta w_i)^3 + \dots \quad (3)$$

The training phase consists roughly in updating the model parameters after the presentation of a single training pattern (on-line mode) or batches of the training set (off-line or batch mode). Each update of model parameters can be viewed as perturbations (e.g. noise) around the current point given by the m dimensional vector of model parameters. Let us assume a given sequence of N disturbance vectors ΔW . Ignoring third and higher order terms in expression (3), the expectation of the error $\langle E(W) \rangle$ can be expressed as:

$$\langle E(W) \rangle \equiv \frac{1}{N} \sum_{n=1}^N E(W + \Delta W^n)$$

$$\langle E(W) \rangle = E(W) + \sum_{i=1}^m \frac{\partial E(W)}{\partial w_i} \frac{1}{N} \sum_{n=1}^N \Delta w_i^n + \frac{1}{2} \sum_{i=1}^m \frac{\partial^2 E(W)}{\partial w_i^2} \frac{1}{N} \sum_{n=1}^N (\Delta w_i^n)^2 + \sum_{i < j} \frac{\partial^2 E(W)}{\partial w_i \partial w_j} \frac{1}{N} \sum_{n=1}^N \Delta w_i^n \Delta w_j^n \quad (4)$$

Rearranging the previous expression we obtain a series expansion of the expectation of the error in terms of the moments of the random perturbations:

$$\langle E(W) \rangle \equiv E(W) + \sum_{i=1}^m \text{mean}(\Delta w_i) \frac{\partial E(W)}{\partial w_i} + \frac{1}{2} \sum_{i=1}^m \text{var}(\Delta w_i) \frac{\partial^2 E(W)}{\partial w_i^2} + \sum_{i < j} \text{cov}(\Delta w_i, \Delta w_j) \frac{\partial^2 E(W)}{\partial w_i \partial w_j} \quad (5)$$

In addition, the weight increment associated to the gradient descent rule is $\Delta w_i = -\eta g_i$. The third term of expression (5) concerning the covariance can be ignored supposing that the elements of the disturbance vectors are uncorrelated over the index n . This is a plausible hypothesis given that patterns are presented randomly to the network during the learning phase. Moreover, close to a local minimum we can assume that $\text{mean}(\Delta w_i) \approx 0$ (the analysis of this approximation is omitted for brevity, but they show that its effects can be neglected). Taking into account these considerations the expectation of the error is then given by:

$$\langle E(W) \rangle \equiv E(W) + \frac{1}{2} \sum_{i=1}^m \sigma^2(\Delta w_i) \frac{\partial E(W)}{\partial w_i^2} = E(W) + \frac{\eta^2}{2} \sum_{i=1}^m \sigma^2(g_i) \frac{\partial^2 E(W)}{\partial w_i^2} \quad (6)$$

From equation (6) it is easy to deduce that the expected value of the error increases as the variance (represented by the symbol σ^2) of the disturbance

vectors (gradients) increases. This observation suggests that the error function should be strongly penalized around such weights. Therefore, to cancel the noise term in the expected value of the error function the gradient descent rule must be changed to $\Delta w_i = -\eta g_i / \sigma g_i$. This normalization is known as vario-eta and was proposed [19] for static networks. Specifically, the learning rate is renormalized by the stochasticity of the error signals. In this line, it is important to note that the error signals contain less and less information at the extent they pass through a long sequence of layers, due to the shrinking procedure carried out by the non-linear (hyperbolic tangent or sigmoidal functions) network units. This is a well-known problem [20] that makes very difficult to find long-term dependencies that could eventually appear encoded within the structural patterns of the training set. For the case of recursive neural networks this problem is even worst due to the unfolding procedure (error signals can traverse many replicas of the same network). However, this normalization procedure avoids the gradients vanishing thanks to the scaling of network weights.

3.2 Algorithm Description

The whole algorithmic description is provided in figure 2. By inspection of the pseudo-code, the algorithm proceeds as follows: after the initialization of algorithm parameters (lines 1 and 2), the algorithm enters in two nested loops: The external or control loop is in charge of monitoring the performance of the algorithm.

In addition, it also performs the update of model parameters (line 12) following the derived learning rule ($\Delta w_i = -\eta g_i / \sigma g_i$). The constant ϕ is summed to the standard deviation of the error gradients for avoiding eventual numerical problems.

Figure 3 shows the pseudo-code of the function S_Gradients (S stands for Structure). This function is in charge of computing the first derivatives of the error function with respect to model parameters. This function takes as arguments a structural pattern U together with its category Y and returns the error gradient vector. The details of the notation used can be found in [22].

The internal loop (lines 5 up to 11) operates recursively in order to obtain the variance (line 9) of the error gradients (line 7). Similarly, the mean value of the gradients is computed recursively in line 8. It is important to note that the algorithm can operate in both batch mode $k_{\max} = \|D\|$ (number of training patterns) or in on-line mode $k_{\max} << \|D\|$ just by selecting appropriately the value of k_{\max} . Finally, it is important to note that the proposed algorithm can be easily adapted to other extensions of the recursive model like contextual models or graph neural networks.

3.3 Preliminary Complexity Analysis

From a computational point of view, the proposed algorithm scales $O(W)$ in terms of memory storage requirements, where $W = W_f \cup W_g$ is the number of parameters of the model. In addition, the computational cost scales roughly as

$O(NW)$ where N is the number of patterns in the data set. In this line, it must be noted that quasi-Newton methods that builds iterative approximations of the inverse Hessian lead in general, to algorithms with overall computational cost of $O(NW^2)$ and memory storage requirements of $O(W^2)$. Similarly, conjugate-gradients method achieves a memory storage of $O(W)$ at the same computational cost of a quasi-Newton method. In addition, it is important to note that a rigorous complexity analysis would require a careful study of the statistical distribution of the gradient errors throughout the optimization procedure followed by an analysis of the generating functions associated to the recursions (lines 8 and 9 of pseudo-code of figure 2).

Therefore, taking into account that the proposed learning rule behaves like a stochastic approximation of a quasi-Newton method, the proposed algorithm achieves a good trade-off in terms of memory storage and computational complexity.

Vario-Eta Through the Structure

Notation:

$$\mathbf{g} = [g_1 \ g_2 \ g_3 \dots \ g_m] \quad \sqrt{\mathbf{g}} = [\sqrt{g_1} \ \sqrt{g_2} \ \dots \ \sqrt{g_m}] \quad \text{diag}(\mathbf{g}^t \mathbf{g}) = \begin{pmatrix} g_1^2 & 0 & \dots & 0 \\ 0 & g_2^2 & \dots & 0 \\ 0 & 0 & \dots & g_m^2 \end{pmatrix}$$

$$a_k = 1 - \frac{1}{k}, \quad b_k = a_k(1 - a_k), \quad \mathbf{l}_m = [1 \ 1 \ 1 \ \dots \ 1], \quad \mathbf{0}_m = [0 \ 0 \ \dots \ 0]$$

Initialize Model Parameters:

- 1: $W = W^f \cup W^g \leftarrow \text{random}, \quad n \leftarrow 0, \quad \eta \leftarrow \eta_0, \quad \phi \leftarrow \phi_0$
- 2: $W = [w_1^f, w_2^f, \dots, w_k^f, w_1^g, w_2^g, \dots, w_l^g], \quad g_0 = \mathbf{0}_m, \quad S_0 = \mathbf{0}_m, \quad (\text{where } m = k+l)$
- 3: **repeat**
- 4: $k \leftarrow 1$
- 5: **repeat**
- 6: randomly select a pattern (U, Y) from training set D
- 7: $\mathbf{g}_k \leftarrow S_{-}\text{Gradients}(U_k, Y_k)$
- 8: $\hat{\mathbf{g}}_k \leftarrow a_k \hat{\mathbf{g}}_{k-1} + (1 - a_k) \mathbf{g}_k$
- 9: $S_k^t \leftarrow a_k S_{k-1}^t + b_k \text{diag}[(\mathbf{g}_k - \hat{\mathbf{g}}_{k-1})^t (\mathbf{g}_k - \hat{\mathbf{g}}_{k-1})] \mathbf{l}_m^t$
- 10: $k \leftarrow k + 1$
- 11: **until** $k = K_{\max}$ (all patterns or a small batch in D have been selected)
- 12: $W_{n+1}^t \leftarrow W_n^t + \text{diag}^{-1} \left[(\phi \mathbf{l}_m + \sqrt{S_{K_{\max}-1}})^t \mathbf{l}_m \right] \hat{\mathbf{g}}_{K_{\max}-1}^t$
- 13: $n \leftarrow n + 1$
- 14: **until** performance criterion is met

Fig. 2. Pseudo-code of the proposed approximated second order stochastic learning algorithm

```

S_Gradients(U,Y)
Choose a reverse topological ordering  $\succ$  on U
for each node  $u \in U$  obeying the ordering  $\succ$  do
     $u.a \leftarrow f(ch[u], I(u))$ 
endfor
for each  $u \in U$  do
     $u.\delta^f \leftarrow 0$ 
     $G^f \leftarrow 0, G^g \leftarrow 0$ 
    Choose a topological ordering on  $\prec$  U
    for each node  $u \in U$  obeying the ordering  $\prec$  do
         $\delta^g \leftarrow y(u.a) - t(u)$ 
         $G^g \leftarrow G^g + J_w^g(u.a)\delta^g$ 
         $u.\delta^f \leftarrow u.\delta^f + J_x^g(u.a)\delta^g$ 
         $G^f \leftarrow G^f + J_w^f(a(ch[u]), I(u)) u.\delta^f$ 
        for  $z \in ch[u]$  do
             $r \leftarrow ord(z, ch[u])$ 
             $z.\delta^f \leftarrow z.\delta^f + J_{x(r)}^f(a(ch[u]), I(u)) u.\delta^f$ 
        endfor
    endfor
return  $G = G^f \cup G^g$ 

```

Fig. 3. Pseudo-code of the function in charge of computing the first derivatives of the error with respect to model parameters

4 Experimental Results

The performance of the algorithm was tested for the problem exposed in [21]. Specifically, this application comes from the intelligent transportation research field and consists on the development of an advanced intersection safety system. The ultimate goal is to provide appropriate warnings to the driver to avoid fatal collisions. For this task, the structural patterns are trees ranging from one up to sixteen levels depth encoding temporal situations at road intersections. Generally speaking, a road intersection situation is composed of a set of dynamic (eg: vehicles, pedestrians, traffic lights, etc) and static entities (eg: trees, bushes, road signs, etc) interacting during a variable time frame. For this application, the pattern set is composed of 4000 structural patterns where approximately half of them representing highly-risky situations (e.g.: situations leading to collisions). The dynamic aspects of road intersections are encoded within the topology of the trees while static aspects are encoded within the label space of the tree-graphs. This task provides an illustrative example of how an extremely complex problem is modeled using the recursive paradigm.

Figure 4 provides a comparison of Back-propagation through the structure (Bpts), a quasi-Newton through the structure algorithm [21] (Qnts) and the proposed algorithm (Vets) running in batch mode. Graphics depict the result of averaging 10 simulations for two different network architectures. Specifically,

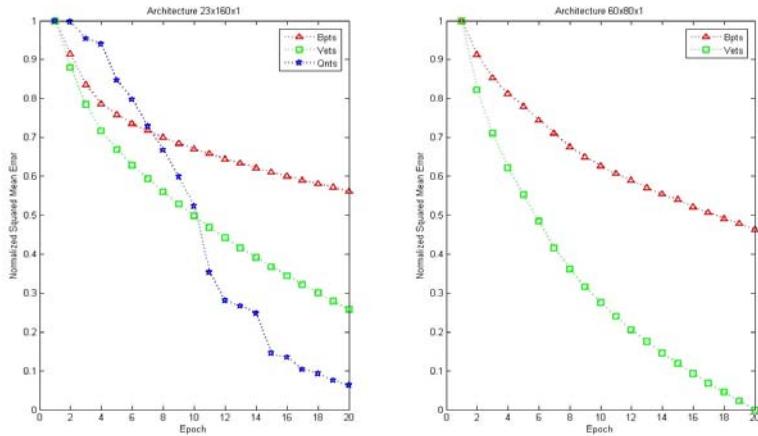


Fig. 4. Simulation results for the proposed stochastic algorithm (Vets), back-propagation through the structure (Bpts) and a quasi-Newton algorithm (Qnts)

each simulation consist on running a training phase of 20 epochs for each algorithm starting from identical weight initialization conditions. Afterwards, the resulting error values are normalized for the three algorithms in the interval [0,1] (the value that is mapped to 0 is the minimum value reached at epoch 20 by the best performing algorithm).

Finally, the normalized error values are averaged over the 10 simulations. The left side of the figure shows the averaged results for an architecture of 23x160x1 (23 units implementing the state transition function and 161 units for the output function). Similarly, the right side of the picture shows the results for a 60x80x1 architecture. Due to the prohibitive memory requirements of the quasi-Newton algorithm (Hessian matrix contains more than 10^7 elements) the comparison was only possible with the Bpts algorithm for this network architecture.

Therefore, although further experimentation must be carried out the proposed stochastic algorithm provides a good trade-off between the memory storage requirements and algorithm complexity.

5 Conclusions

In this paper we have described the principles behind the recursive neural network model. It was shown that associated with any given problem, the information content presents a certain geometry that these models can attempt to exploit. Furthermore, the fact of using structured representations of information is translated into a substantial gain in information content. In addition, in order to tackle the inherent complexity of these models a stochastic learning algorithm was also described. The proposed algorithm is able to achieve a good trade-off between speed of convergence and the computational effort required by setting

the local learning rate for each weight inversely proportional to the standard deviation of its stochastic gradient. The scaling properties of the algorithm make it suitable for the computational requirements of the recursive model. Furthermore, the proposed learning scheme can be easily adapted to other recursive models such as contextual models or graph neural networks. The computer simulations demonstrated the efficiency of the algorithm for a practical learning task.

References

1. Goller, C., Kuchler, A.: Learning Task-Dependent Distributed Structure-Representations by Backpropagation Through Structure. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN 1996), Washington, pp. 347–352 (1996)
2. Frasconi, P., Gori, M., Sperduti, A.: A General Framework for Adaptive Processing of Data Structures. *IEEE Transactions on Neural Networks* 9(5), 768–786 (1998)
3. Ceroni, A., Frasconi, P., Pollastri, G.: Learning Protein Secondary Structure From Sequential and Relational Data. *Neural Networks* 18, 1029–1039 (2005)
4. Baldi, P., Pollastri, G.: The Princpled Design of Large-Scale Recursive Neural Networks Architectures-DAG-RNNs and the Protein Structure Prediction Problem. *Journal of Machine Learning Research* 4, 575–602 (2003)
5. Mauro, C.D., Diligenti, M., Gori, M., Maggini, M.: Similarity Learning for Graph-based Image Representations. *Pattern Recognition Letters* 24(8), 1115–1122 (2003)
6. Costa, F., Frasconi, P., Lombardo, V., Soda, G.: Towards Incremental Parsing of Natural Language Using Recursive Neural Networks. *Applied Intelligence* 19, 9–25 (2003)
7. Bianchini, M., Gori, M., Sarti, L., Scarselli, F.: Recursive Processing of Cyclic Graphs. *IEEE Transactions on Neural Networks* 9(17), 10–18 (2006)
8. Gori, M., Monfardini, G., Scarselli, L.: A New Model for Learning in Graph Domains. In: Proceedings of the 18th IEEE International Joint Conference on Neural Networks, Montreal, pp. 729–734 (2005)
9. Hammer, B., Micheli, A., Sperduti, A.: Adaptive Contextual Processing of Structured Data by Recursive Neural Networks: A Survey of Computational Properties. In: Hammer, B., Hitzler, P. (eds.) *Perspectives of Neural-Symbolic Integration*. Springer, Heidelberg (2007)
10. Micheli, A., Sona, D., Sperduti, A.: Contextual Processing of Structured Data by Recursive Cascade Correlation. *IEEE Transactions on Neural Networks* 15(6), 1396–1410 (2004)
11. Hammer, B., Micheli, A., Sperduti, A.: Universal Aproximation Capabilities of Cascade Correlation for Structures. *Neural Computation* 17, 1109–1159 (2005)
12. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support Vector Machine Learning for Interdependent and Structured Output Spaces. In: Brodley, C.E. (ed.) *ICML 2004: Twenty-first international conference on Machine Learning*. ACM Press, New York (2004)
13. Hammer, B., Saunders, C., Sperdutti, A.: Editorial of the Special issue on Neural Networks and Kernel Methods for Structured Domains. *Neural Networks* 18(8), 1015–1018 (2005)
14. Leyton, M.: A Generative Theory of Shape. LNCS, vol. 2145, pp. 1–76. Springer, Heidelberg (2001)
15. Leyton, M.: Symmetry, Causality, Mind. MIT Press, Massachusetts (1992)

16. Churchland, P., Sejnowski, T.: *The Computational Brain*. MIT Press, Cambridge (1992)
17. Hecht-Nielsen, R.: *Confabulation Theory: The Mechanism of Thought*. Springer, Heidelberg (2007)
18. Baldi, P., Rosen-Zvi, M.: On the Relationship between Deterministic and Probabilistic Directed Graphical Models: from Bayesian Networks to Recursive Neural Networks. *Neural Networks* 18(8), 1080–1086 (2005)
19. Orr, G.B., Müller, K.-R. (eds.): *NIPS-WS 1996*. LNCS, vol. 1524, pp. 395–399. Springer, Heidelberg (1998)
20. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: A Field Guide to Dynamical Recurrent Networks. In: Kolen, J., Kremer, S. (eds.), pp. 237–243. IEEE Press, Inc., New York (2001)
21. Chinea, A., Parent, M.: Risk Assessment Algorithms Based on Recursive Neural Networks. In: Proceedings of the 20th IEEE International Joint Conference on Neural Networks, Florida, pp. 1434–1440 (2007)
22. Frasconi, P., Gori, M., Kuchler, A., Sperduti, A.: A Field Guide to Dynamical Recurrent Networks. In: Kolen, J., Kremer, S. (eds.), pp. 351–364. IEEE Press, Inc., New York (2001)

An EM Based Training Algorithm for Recurrent Neural Networks

Jan Unkelbach, Sun Yi, and Jürgen Schmidhuber

IDSIA, Galleria 2, 6928 Manno, Switzerland

{jan.unkelbach,yi,juergen}@idsia.ch

<http://www.idsia.ch>

Abstract. Recurrent neural networks serve as black-box models for nonlinear dynamical systems identification and time series prediction. Training of recurrent networks typically minimizes the quadratic difference of the network output and an observed time series. This implicitly assumes that the dynamics of the underlying system is deterministic, which is not a realistic assumption in many cases. In contrast, state-space models allow for noise in both the internal state transitions and the mapping from internal states to observations. Here, we consider recurrent networks as nonlinear state space models and suggest a training algorithm based on Expectation-Maximization. A nonlinear transfer function for the hidden neurons leads to an intractable inference problem. We investigate the use of a Particle Smoother to approximate the E-step and simultaneously estimate the expectations required in the M-step. The method is demonstrated for a synthetic data set and a time series prediction task arising in radiation therapy where it is the goal to predict the motion of a lung tumor during respiration.

Keywords: Recurrent neural networks, Dynamical System identification, EM, Particle Smoother.

1 Introduction

Recurrent neural networks (RNNs) represent dynamical systems. With a large enough number of hidden neurons, a given dynamical system can in principle be approximated to arbitrary precision. Hence, recurrent networks have been used as black-box models for dynamical systems, e.g. in the context of time series prediction [1].

When using recurrent neural networks for dynamical system identification, it is typically assumed that the dynamics of the system to be modeled is deterministic. RNNs do not model *process noise*, i.e. uncertainty in the internal state transition between two time steps. Hence, RNNs may not be suitable models for systems exhibiting process noise as a characterizing feature.

In contrast, state space models allow for noise in both the internal state transitions and the mapping from internal states to observations, and therefore represent a richer class of probabilistic models for sequential data. In the special

case of linear-gaussian state space models [2, chapter 13.3], the parameters of the model can be learned using an instantiation of the Expectation-Maximization (EM) algorithm [3]. However, computationally tractable algorithms for exact inference in the E-step are limited to the linear-gaussian case. Nonlinear, non-gaussian models require approximation.

Here, we consider RNNs as nonlinear state space models and suggest a training algorithm based on Expectation-Maximization. We investigate the use of a sequential sampling method, the Particle Smoother [4], to approximate the E-step and simultaneously estimate the expectations required in the M-step. In the M-step, the similarity of linear-gaussian state space models and simple recurrent networks can be exploited.

The proposed method has the advantage that the RNN can model dynamical systems characterized by process noise, and that it represents a *generative* model of the data. This is not the case for conventionally trained RNNs, e.g. using a quadratic objective function minimized through gradient descent.

Stochasticity in recurrent networks has been introduced before in the context of training algorithms based on Extended Kalman Filtering [5]. However, this work did not aim at training a recurrent network as a generative model for a stochastic dynamical system, but at further developing training algorithms for conventional recurrent networks.

Related work includes inference and learning in nonlinear state space models in general. For example, Ghahramani [6] addresses learning in nonlinear state space models where the nonlinearity is represented by radial basis functions. The E-step is approximated via Extended Kalman Filtering. The M-step can be solved in closed form for this model.

Our algorithm is demonstrated for a synthetic data set and a time series prediction task arising in radiation therapy. Lung tumors move during respiration due to expansion and contraction of the lungs. Online adjustment of the radiation beam for tumor tracking during treatment requires the prediction of the tumor motion for about half a second. The variability of the breathing pattern is characterized by process noise rather than measurement noise.

The remainder of this paper is organized as follows: Section 2 reviews Linear Dynamical Systems¹ and Recurrent neural networks as black-box models for dynamical systems. Section 3 describes the EM-based training algorithm for recurrent networks, where subsection 3.1 details the sampling method used in the E-step. Section 4 discusses two applications of the method.

2 Dynamical System Identification

Given is a sequence $Y = \{y_t\}_{t=1}^T$, where y_t is a vector of observations at time step t . We wish to find a model of the underlying dynamical system that generated the sequence, e.g. for the purpose of time series prediction. Below, we introduce

¹ Linear Dynamical System and Linear-gaussian state space models refer to the same model.

notation, review linear dynamical systems (LDS) and recurrent neural networks, and address the parameter learning problem in these models.

2.1 Linear Dynamical Systems

A stochastic linear dynamical system is described by

$$x_t = Ax_{t-1} + \eta_t^p \quad (1)$$

$$y_t = Bx_t + \eta_t^m \quad (2)$$

where x is an internal state vector, y is an observation vector, A and B are matrices with constant coefficients, η^p is zero mean gaussian process noise with covariance matrix Γ , and η^m is zero mean gaussian measurement noise with covariance matrix Σ .

The stochastic LDS can be formulated as a probabilistic model with observed variables y_t and latent variables x_t . The model is defined via the conditional probabilities

$$P(x_t|x_{t-1}) = \mathcal{N}(x_t|Ax_{t-1}, \Gamma) \quad (3)$$

$$P(y_t|x_t) = \mathcal{N}(y_t|Bx_t, \Sigma) \quad (4)$$

Hence, the hidden variables x_t form a markov chain.

2.2 Learning in Linear Dynamical Systems

In LDS the model parameters $\theta = (A, B, \Gamma, \Sigma)$ can be learned using maximum likelihood through an instantiation of the EM algorithm [2, chapter 13.3.2]. In the E-step, we need to solve the inference problem and calculate the marginal probabilities of the hidden variables conditioned on the observation sequence:

$$P(X|Y; \theta) \quad (5)$$

where $X = \{x_t\}_{t=1}^T$ and $Y = \{y_t\}_{t=1}^T$ denote the entire sequence of hidden states and observations, respectively. For LDS, the inference problem can be solved exactly as $P(X|Y; \theta)$ remains a Gaussian. Its mean and covariance matrix can be calculated with a forward-backward algorithm. Given the posterior distributions over the latent variables, the M-step can also be performed analytically.²

2.3 Recurrent Neural Networks

If we model the sequence of observations with a recurrent neural network³, equations 1 and 2 are replaced by

$$x_t = Af(x_{t-1}) + \eta_t^p \quad (6)$$

$$y_t = Bf(x_t) + \eta_t^m \quad (7)$$

² Here, we assume for simplicity that $x_0 = 0$. However, it is straight forward to learn the mean and covariance matrix of a Gaussian distribution over the initial internal state.

³ This type of network is also referred to as an Elman network.

where x is now interpreted as the vector of net inputs of the hidden neurons, $f = \tanh$ is the transfer function of the hidden neurons so that $f(x)$ is the vector of hidden neuron activations, y is the network output, A is the weight matrix for the recurrent connections, and B is the output weight matrix. Hence, the structure of the RNN model is very similar to LDS. The measurement equation 7 represents a linear output layer. The network has no external inputs in this formulation (except the process noise which can be interpreted as an unknown external influence).

2.4 Conventional Learning in Recurrent Neural Networks

When training recurrent networks, the process noise η^p is typically neglected. In addition, the measurement noise η^m is assumed to be uncorrelated and of the same variance in all components of y . In this case, maximizing the likelihood of the data corresponds to minimizing a quadratic cost function:

$$\underset{A,B}{\text{minimize}} \quad \frac{1}{2} \sum_{t=1}^T [y_t - Bf(x_t)]^2 \quad (8)$$

Minimization of the cost function can be performed via gradient descent. Back-propagation through time (BPTT) and Real time recurrent learning (RTRL) are well-known algorithms to calculate the gradient 7. Alternative methods include Evolino 8, where the matrix A is determined via evolutionary methods whereas B is determined through linear regression. In Echo-State networks 9, A is chosen to be a fixed sparse random matrix and B is determined via linear regression.

3 EM Based Learning in Recurrent Neural Networks

Here, we wish to train recurrent networks without neglecting the process noise term. The RNN is considered as a nonlinear state-space model. We exploit the similarities of RNNs and LDS to obtain an EM-based training algorithm.

We wish to maximize the following likelihood function with respect to the model parameters $\theta = (A, B, \Gamma, \Sigma)$:

$$\underset{\theta}{\text{maximize}} \quad L = \int P(X, Y | \theta) dX \quad (9)$$

where $\int dX$ denotes the integration over all latent variables $X = \{x_t\}_{t=1}^T$. The complete data Likelihood is given by

$$P(X, Y | \theta) = \prod_{t=1}^T P(x_t | x_{t-1}; A, \Gamma) P(y_t | x_t; B, \Sigma) \quad (10)$$

so that the complete data log-Likelihood is given by

$$\ln P(X, Y | \theta) = \sum_{t=1}^T \ln \mathcal{N}(x_t | Af(x_{t-1}), \Gamma) + \sum_{t=1}^T \ln \mathcal{N}(y_t | Bf(x_t), \Sigma) \quad (11)$$

In the M-step, we maximize the following Q-function with respect to the model parameters θ :

$$\underset{\theta}{\text{maximize}} \quad Q(\theta|\theta^{old}) = \int P(X|Y; \theta^{old}) \ln P(X, Y|\theta) dX \quad (12)$$

The structure of the Q-function is, apart from the non-linear transfer function f , identical to the one for linear dynamical systems [2, chapter 13.3.2]. This similarity can be exploited in the M-step as detailed in section 3.2.

In the E-step, we need to determine the joint probability distribution over the latent variables X conditioned on the observation sequence Y , given the current parameter values θ^{old} :

$$P(X|Y; \theta^{old}) \quad (13)$$

Since exact inference is intractable, we employ a sampling method, the Particle Smoother, to approximate the distribution 13 as detailed in section 3.1.

3.1 The Particle Smoother for Approximate Inference

In the E-step of the Expectation-Maximization algorithm, we wish to approximate the probability density of the latent variables X conditioned on the observed sequence Y . However, inspecting the structure of the log-Likelihood function 11 shows that we only need the marginals $P(x_t|Y; \theta)$ and $P(x_t, x_{t-1}|Y; \theta)$.

To start, we consider $P(x_t|Y_t)$, i.e. the probability of x_t given the observations $Y_t = \{y(t')\}_{t'=1}^t$ until time step t . This can be approximated via a sequential Monte Carlo method, the particle filter. At each time step t , the distribution is approximated by

$$\hat{P}(x_t|Y_t) = \sum_{i=1}^N w_t^i \delta(x_t - x_t^i) \quad (14)$$

where x_t^i is the position of particle i at time t , w_t^i is the particle weight, δ denotes the delta-function, and N is the number of particles. To proceed to the next time step, we sample new particles x_{t+1}^j from a mixture of Gaussians:

$$x_{t+1}^j \sim \sum_{i=1}^N w_t^i \mathcal{N}(x_{t+1}^j | Af(x_t^i), \Gamma) \quad (15)$$

The new particles are weighted according to the probability of generating the next obsevation y_{t+1} :

$$w_{t+1}^j = \frac{P(y_{t+1}|x_{t+1}^j)}{\sum_{k=1}^N P(y_{t+1}|x_{t+1}^k)} \quad (16)$$

An approximation of the probability density $P(x_t|Y)$, now conditioned on the entire observation sequence Y , can be obtained using a forward-backward smoother 4. Following this method, we first approximate $P(x_t|Y_t)$ for every time step using

a particle filter, and in a second step, correct the particle weights via a backwards recursion. The smoothed distribution $P(x_t|Y)$ is then approximated via

$$\hat{P}(x_t|Y) = \sum_{i=1}^N v_t^i \delta(x_t - x_t^i) \quad (17)$$

with modified weights v_t^i . To derive the backward recursion, we consider the following identity:

$$\begin{aligned} P(x_t|Y) &= \int P(x_{t+1}|Y) P(x_{t+1}|x_t, Y) dx_{t+1} \\ &= \int P(x_t|Y_t) \frac{P(x_{t+1}|Y) P(x_{t+1}|x_t)}{\int P(x_{t+1}|x_t) P(x_t|Y_t) dx_t} dx_{t+1} \end{aligned} \quad (18)$$

By inserting equations 14 and 17 into 18, we obtain the backward recursion for the corrected weights v_t^i :

$$v_t^i = w_t^i \left[\sum_{j=1}^N v_{t+1}^j \frac{P_t^{ji}}{\sum_{k=1}^N w_t^k P_t^{jk}} \right] \quad (19)$$

where $P_t^{ji} = P(x_{t+1}^j|x_t^i)$ is the transition probability from particle i at time t to particle j at time $t+1$. The backward recursion is initialized as $v_T^i = w_T^i$. After obtaining particle weights w_t^i in the forward pass and weights v_t^i in the backward pass, equation 18 gives also rise to an approximation of the joint probability for x_t and x_{t+1} :

$$\hat{P}(x_t, x_{t+1}|Y) = \sum_{i=1}^N \sum_{j=1}^N \left[\frac{w_t^i v_{t+1}^j P_t^{ji}}{\sum_{k=1}^N w_t^k P_t^{jk}} \delta(x_t - x_t^i) \delta(x_{t+1} - x_{t+1}^j) \right] \quad (20)$$

3.2 The M-Step

In the M-step, we need to maximize the Q-function 12 with respect to the model parameters $\theta = (A, B, \Gamma, \Sigma)$. Here, we can make use of the fact that the structure of the dynamic equation describing the RNN is similar to those for LDS. This leads to similar update equations for the parameters. Exemplarily, we consider the update equation for the recurrent weight matrix A which we obtain by setting the Q-function derivative with respect to A to zero:

$$A^{new} = \left[\sum_{t=1}^T \mathbb{E}[x_t f(x_{t-1})^\top] \right] \left[\sum_{t=1}^T \mathbb{E}[f(x_{t-1}) f(x_{t-1})^\top] \right]^{-1} \quad (21)$$

where \mathbb{E} denotes the expectation with respect to the distribution $P(X|Y; \theta^{old})$. The expectations in equation 21 are approximated through the particle smoother as

$$\mathbb{E} [f(x_{t-1})f(x_{t-1})^\top] = \sum_{i=1}^N v_{t-1}^i f(x_{t-1}^i) f(x_{t-1}^i)^\top \quad (22)$$

$$\mathbb{E} [x_t f(x_{t-1})^\top] = \sum_{i=1}^N \sum_{j=1}^N \frac{w_{t-1}^i v_t^j P_{t-1}^{ji}}{\sum_{k=1}^N w_t^k P_{t-1}^{jk}} x_t^i f(x_{t-1}^i)^\top \quad (23)$$

The update equations for the parameters B , Γ and Σ are obtained in analogy to the update equations for linear-gaussian models as described in [2 chapter 13.3.2]. They are omitted here due to space limitations.

4 Applications

The method is demonstrated for a synthetic data set and a real data set describing irregular breathing motion of a lung cancer patient.

4.1 Synthetic Data

We consider synthetic data generated by a recurrent network with two hidden neurons with parameters

$$A = \begin{pmatrix} 1 & 1 \\ -0.05 & 0.98 \end{pmatrix} \quad B = (1 \ 0) \quad \Gamma = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.001 \end{pmatrix} \quad \Sigma = (0.01)$$

These parameters were chosen so that the system outputs an oscillation which is perturbed by both process and measurement noise. Without noise, the system generates an (almost harmonic) oscillating output signal with a period of 28 time steps. Figure 1 shows a sample sequence generated by the system with noise. A recurrent network with 5 hidden neurons⁴ was trained on a data set containing 1000 time steps. Figure 1 shows parts of a test data set, together with predictions of the trained RNN (blue circles). Every 20 time steps, the network prediction for the next 10 time steps is shown. For prediction, a particle filter is used to estimate the mean of the posterior distribution over the hidden state. Then the deterministic dynamics of the network (without noise) is used for prediction, starting from the estimated hidden state⁵. In order to compare the predictions of the RNN, figure 1 also shows the predictions based on the true dynamical system (green dots). In this example, the prediction performance of the RNN is similar to the best possible prediction where the true generative process of the data is known⁶.

Figure 2 shows the internal, noise free dynamics of the network. The network output is shown as the thick red line, whereas the blue thin lines show the

⁴ Similar results are obtained with 2 or more hidden neurons.

⁵ A prediction method where all particles were propagated forward in time and averaged at each time step, yields almost identical results.

⁶ The mean squared errors of the trained network are 0.145/0.259/0.327 for the 1/5/10-step prediction. The corresponding values for the true system are 0.142/0.240/0.307. The standard deviation of the test data is 0.380.

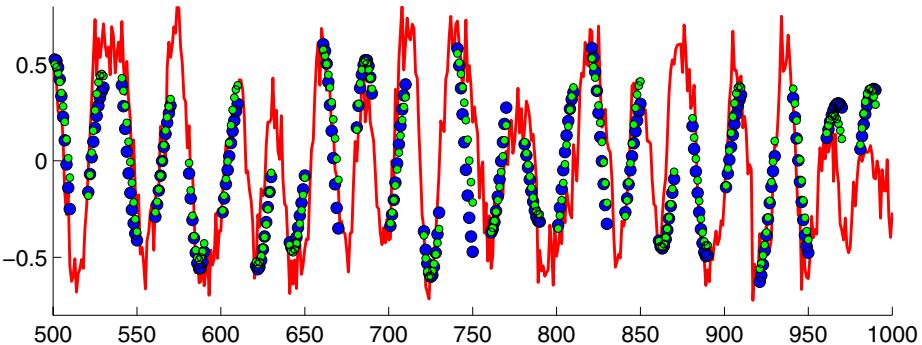


Fig. 1. Synthetic time series (red line). Every 20 time steps, the network predictions for the next 10 time steps are shown (blue circles), together with the predictions obtained using the true dynamical system (green dots).

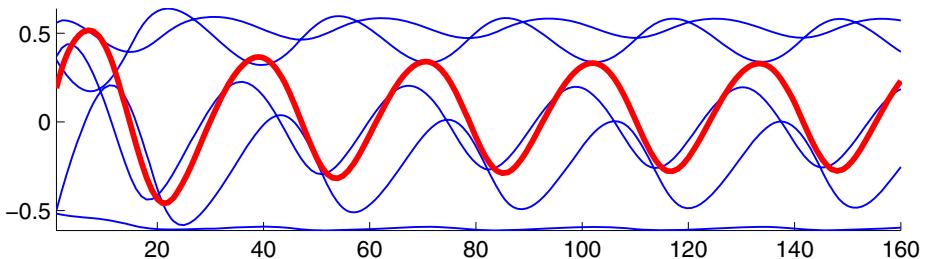


Fig. 2. Deterministic dynamics of the network trained on the noisy data set. Red thick line: network output; Blue thin lines: internal states.

dynamics of the internal state. The network was able to learn the underlying harmonic oscillation of the dynamical system and estimate the amount of process and measurement noise.

4.2 Breathing Motion Data

Figure 3 (red line) shows a surrogate for breathing motion for a lung cancer patient⁷. The signal is roughly periodic, but has substantial variations in amplitude and period. In radiation therapy practice, the goal is to predict the breathing pattern for about 500 milliseconds (5 time steps), where the length of one breathing cycle is around 3 seconds. It is also of interest to have a generative model of the breathing signal in order to simulate the effect of breathing on the accuracy of a radiation treatment.

A recurrent network with 10 hidden neurons was trained on a sequence of 1000 time steps. Figure 4 shows the intrinsic dynamics of the trained network,

⁷ The expansion of the abdomen was measured as a function of time.

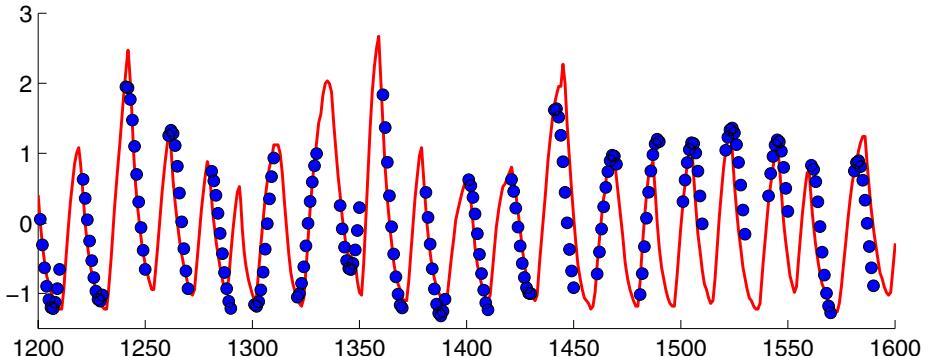


Fig. 3. Breathing signal time series (red line). Every 20 time steps, the next 10 time steps predicted by a trained network are shown (blue circles).

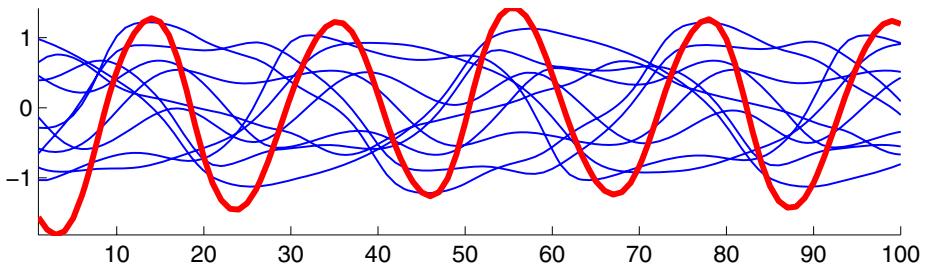


Fig. 4. Deterministic, noise free dynamics of a trained network. Red thick line: network output; Blue thin lines: internal states.

i.e. the dynamics without noise. Figure 3 shows samples of the network prediction on the test data. Every 20 time steps, the network predictions for the next 10 time steps are shown. Like in the synthetic data example above, the training algorithm is able to find solutions that model the oscillatory behaviour of the system. However, it failed to model the dynamics very precisely. Consequently, in order to explain the deviation of data and network output, the estimated amount of measurement noise Σ is too large, and the trained network does not represent an adequate generative model.

4.3 Remarks

Initialization of parameters. The weight matrix A was initialized to a diagonal matrix plus random values with a standard deviation of 0.1. The components of B were initialized to random values with mean one and standard deviation 0.1. These values led to solutions qualitatively similar to those shown above.

Local minima. Dynamical system identification tends to be a difficult task. For the breathing motion data set, the network did not learn details of the dynamics.

However, the same problem applies to linear-gaussian state space models and RNNs trained with gradient descent.

Number of particles. The results presented above were generated with $N = 100$ particles. However, similar results were obtained for only 10 particles – although particle numbers that low are not expected to provide an adequate characterization of the posterior distribution at a given time step. This aspect will be investigated in future work.

5 Conclusion

We address the problem of training recurrent neural networks as black box models for stochastic nonlinear dynamical systems. In conventional training algorithms based on a quadratic objective function, it is assumed that the underlying dynamics to be modelled is deterministic, i.e. process noise is neglected. In this paper, we generalize RNNs to model stochastic dynamical systems characterized by process noise. We suggest a training algorithm based on Expectation-Maximization, exploiting the similarities between RNNs and linear dynamical systems. We apply a particle smoother for approximate inference in the E-step and simultaneously estimate expectations required in the M-step. The algorithm is successfully demonstrated for one synthetic and one realistic data set. Further characterization of the performance of the training algorithm, the influence of the number of particles, and comparison to standard training methods for RNNs are subject to current studies.

Acknowledgment

This research was funded by SNF grant 200021-111968/1.

References

1. Mandic, D.P., Chambers, J.A.: Recurrent neural networks for prediction. John Wiley & Sons, Inc., Chichester (2001)
2. Bishop, C.M.: Pattern recognition and Machine learning. Springer, Heidelberg (2006)
3. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions. John Wiley & Sons, Inc., Chichester (1997)
4. Klaas, M., Briers, M., Doucet, A., Maskell, S.: Fast particle smoothing: If I had a million particles. In: International Conference on Machine Learning (ICML), pp. 25–29 (2006)
5. Williams, R.J.: Training recurrent networks using the extended kalman filter. In: Proceedings International Joint Conference on Neural Networks, pp. 241–246 (1992), citeseer.nj.nec.com/williams92training.html
6. Ghahramani, Z., Roweis, S.T.: Learning nonlinear dynamical systems using an EM algorithm. In: Advances in Neural Information Processing Systems 11, pp. 599–605. MIT Press, Cambridge (1999)

7. Williams, R.J., Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity. In: Back-propagation: Theory, Architectures and Applications. Erlbaum, Hillsdale (1994)
8. Schmidhuber, J., Wierstra, D., Gagliolo, M., Gomez, F.: Training recurrent networks by evolino. *Neural Computation* 19(3), 757–779 (2007)
9. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001)

Modeling D_{st} with Recurrent EM Neural Networks

Derrick Takeshi Mirikitani and Lahcen Ouarbya

Department of Computer Science,
Goldsmiths College, University of London,
New Cross, London, London SE14 6NW, UK
D.T.Mirikitani@gold.ac.uk

Abstract. Recurrent Neural Networks have been used extensively for space weather forecasts of geomagnetospheric disturbances. One of the major drawbacks for reliable forecasts have been the use of training algorithms that are unable to account for model uncertainty and noise in data. We propose a probabilistic training algorithm based on the Expectation Maximization framework for parameterization of the model which makes use of a forward filtering and backward smoothing Expectation step, and a Maximization step in which the model uncertainty and measurement noise estimates are computed. Through numerical experimentation it is shown that the proposed model allows for reliable forecasts and also outperforms other neural time series models trained with the Extended Kalman Filter, and gradient descent learning.

1 Introduction

Neural Networks have established themselves as effective tools in the prediction of geomagnetic disturbances [4][7][8]. In this paper we extend the work in the field of geomagnetic storm forecasting by proposing an *EM* Kalman filtering and smoothing framework for estimation of recurrent neural network (RNN) parameters. The advantage of our approach is the probabilistic representation of model uncertainty and noise in the data, which has been neglected in prior work in the area. This is achieved through the use of the *EM* algorithm for tuning of model hyper-parameters representing the process and measurement noise components, leading to improved out of sample performance on D_{st} forecasting tasks. In the following sub-section we provide an overview of geomagnetospheric disturbances and a brief review of neural forecasting of geomagnetic storms.

1.1 Geomagnetic Storm Forecasting

A number of studies [1][3][5] have shown that changes in the solar activity induces a disturbance of the earth's magnetic field. This disturbance of the near-earth environment is influenced by the injection of energetic particles through the solar wind¹ into the magnetosphere. The solar wind carries the sun's magnetic field through the solar system, forming what we call the Interplanetary Magnetic Field² (*IMF*). A transfer of energy

¹ Solar winds are a stream of charged particles, mostly electrons and protons, that are ejected from the upper atmosphere of the sun.

² The sun's magnetic field carried through the solar system by the solar wind.

from the solar wind into the magnetosphere takes place when the *IMF* opposes the Earth's magnetic field. A disturbance of the magnetosphere, known as a magnetic storm, occurs if this transfer of energy persists for several hours [5]. Geomagnetic storms can have many negative effects on technical systems in space and on Earth, such as a change in a spacecraft orientation or disruption of power lines on the Earth.

Several models have been applied for the prediction of the ring current index D_{st} including feed-forward, FIR, and recurrent neural networks [4][7][8]. Also, various combinations of exogenous variables such as proton density (n), velocity (v), and the *IMF* data (b , b_x , b_y , b_z)³ have been used as inputs to various types of neural networks. It was found that forecasts produced by feed-forward Neural Networks gives good results for the initial and main phase of magnetic storms, but not the recovery phase [5]. This results from the fact that the internal state of the magnetosphere has a great impact on the storm recovery phase. Freeman et al. [4] have used the D_{st} index the magnitude of the *IMF*(b), the *IMF* southward component (b_z), and the solar wind dynamic pressure (n) to give one hour ahead predictions of the D_{st} index. The use of Elman RNNs [9][7] has shown to give superior D_{st} predictions than feed-forward Neural Networks. Wu and Lundstedt [17] have used different coupling functions of solar wind and *IMF* parameters as inputs to their Elman RNN for D_{st} prediction. In [9] Lundstedt et al. have used an optimised RNN, driven solely by hourly averages of the solar wind parameters, particle density (n), and velocity (v) and the *IMF* southward component, b_z . They have shown that their recurrent model has smaller errors than previous models. Palloccia et al. [10] argued that the instruments used to measure the plasma parameter, pressure and velocity, can often be affected by enhanced X-Ray and energetic particle flux. This gives rise to wrong or missing gaps in the plasma data. The use of wrong data can have a negative effect on storm prediction. To circumvent this data quality problem, a model built with *IMF* parameters, b_z , b^2 and b_y^2 , only was proposed to predict D_{st} . It was found that for quiet periods the inputs of Lundstedt and Wu [9] perform better than *IMF* only models, however during severe storms, *IMF* only models provide better predictions. The authors concluded that for the period of severe storms the plasma data are not reliable and will deleteriously affect the predictions of magnetic activity. Recent research has experimented with various advanced machine learning techniques such as wavelet based forecasting [15] and radial basis function networks [16]. The results reported with these alternative methods suggest that significant improvements in forecasts can be achieved with advanced machine learning techniques.

We elaborate on a Maximum Likelihood training framework based on the *EM* algorithm, for RNN parameter estimation, which allows for estimation of model uncertainty and noise in the data [2][6][14]. The extended Kalman Filter (EKF) [11] and smoother are used for sequential estimation of the network weights. The EKF is a second order estimation algorithm for Neural Networks [13], which evolves an approximate covariance matrix that encodes second order information about the underlying system during training. Through the use of the EM algorithm, the process and measurement noise hyper-parameters of the filters are estimated via a maximum likelihood approach, which

³ b is the magnitude of the *IMF*, where, b_x , b_y , and b_z are the three components of the *IMF*.

leads to improved out of sample performance on D_{st} forecasting. Experimental simulations show that forecasts of D_{st} with the EM-RNN algorithm provide more accurate forecasts than the RNN trained with first-order gradient descent [18], and an RNN trained with the second-order extended Kalman filter(EKF) [11].

This paper is organised as follows. In Section 1.1 the architecture of the RNN network is described. The next section provides the extended Kalman filter and smoother. Section 4 offers the EM-RNN training approach for recursive hyperparameter re-estimation. Section 5 presents the simulation results, and finally a brief conclusion is provided in Section 6.

2 Recurrent Neural Networks

RNNs are nonlinear adaptive models with internal states trainable by specialised weight adaptation algorithms. The recurrent architecture chosen for this study is known as the Williams and Zipser fully recurrent network [18], which is the most general RNN architecture (shown in figure 1). We adopt the following notation to describe the fully recurrent network: \mathbf{s}_t is the input vector for each neuron, which contains the exogenous input to the network x_t , the bias b , and the previous activation of each neuron $\mathbf{s}_t = [c^{(1)}, \dots, c^{(H)}, x^{(H+1)}, b^{(H+2)}]$, where $\{c^{(1)}, c^{(2)}, \dots, c^{(H)}\}$ are the activations of the network at the previous time step. The superscript (l) refers to the l^{th} element $s^{(l)} \in \mathbf{s}_t$ for $l = 1, 2, \dots, H + 2$ where H is the number of neurons. The output activation of each neuron is defined as a function $y_t^{(i)} = g(\mathbf{w}_t^{(i)}, \mathbf{s}_t)$ where $\mathbf{w}_t^{(i)} = [w_{i,1}, \dots, w_{i,H+2}]$ is the weight vector associated with the i^{th} neuron at time t , and the overall network weight vector is defined as $\mathbf{W} = [\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(H)}]$. The functions $g(\cdot)$ are logistic sigmoidal nonlinearities $g(a) = 1/(1 + \exp(-a))$ which map the input a from \mathcal{R} into a bounded interval $\Omega = (0, 1)$ of length $|\Omega| = 1$ where $\Omega \subset \mathcal{R}$.

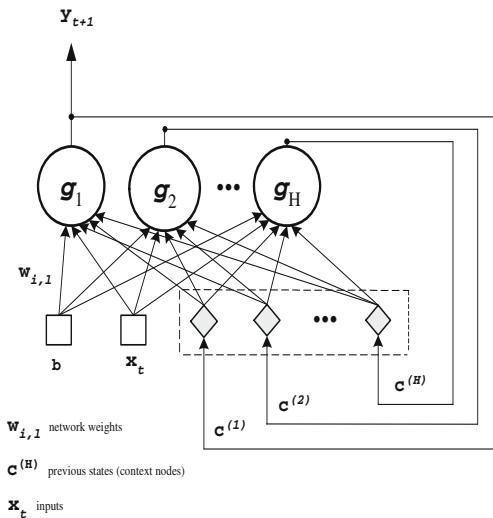


Fig. 1. Fully Recurrent Neural Network

The fully recurrent Neural Network architecture consists of a single layer of processing neurons which are fully connected to each other. The input layer consists of the exogenous inputs along with the bias, which feed temporal information to the processing layer at each time step. Each neuron in the processing layer computes a weighted sum of the previous processing layer activations, along with the exogenous inputs to the network, and the bias given by

$$v_t^{(i)} = \sum_{l=1}^{H+2} w_{i,l} s_t^{(l)} \quad (1)$$

where $w_{i,l}$ is the weight connecting the i^{th} neuron to the l^{th} component of the \mathbf{s} vector. Each weighted sum v_i is then passed through the nonlinear activation function to produce the activation outputs

$$y_t^{(i)} = g^{(i)}(v_t^{(i)}) \quad (2)$$

where the output of the network is y_1 . The entire network is referred to as highly nonlinear function h of the weights \mathbf{W}_t and input x_t

$$d_t = h(\mathbf{W}_t, x_t) + \epsilon_t \quad (3)$$

where the noise ϵ_t is assumed to be independent zero-mean Gaussian with covariance R : $\epsilon_t \sim \mathcal{N}(0, R)$, and where d_t are the targets from the provided data set $D = \{x_t, d_t\}_{t=1}^N$, where $x_{t+1} = d_t$.

3 Kalman Filtering and Smoothing

One of the most popular recursive Bayesian state estimation algorithms for RNN training is the extended Kalman filter (EKF). In the Kalman filtering framework, it is assumed that the Neural Network parameter vector evolves over time in the sense of a first order stochastic process

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \boldsymbol{\nu}_t \quad (4)$$

where the process noise is assumed to be normally distributed $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{w}_{t-1}, \mathbf{Q})$. The RNN provides a nonlinear mapping of the evolving state and the system inputs to the measurements y_t , as defined in equation [3]

For Neural Networks, the EKF requires the computation of the Jacobian matrix $\mathbf{j}_t = \partial h(\cdot) / \partial \mathbf{W}_t$ of partial derivatives of the output y_{t+1} with respect to the weights of the network, where the Jacobian \mathbf{j}_t is evaluated at each time step, usually with the RTRL algorithm [18]. The following equations describe the EKF training algorithm for the RNN:

$$\begin{aligned} \mathbf{W}_{t+1} &= \mathbf{W}_t \\ \mathbf{P}_{t+1}^t &= \mathbf{P}_t^t + \mathbf{Q} \\ \mathbf{K}_{t+1}^{t+1} &= \mathbf{P}_{t+1}^t \mathbf{j}_{t+1} [\mathbf{j}_{t+1} \mathbf{P}_{t+1}^t \mathbf{j}_{t+1}^T + R]^{-1} \\ \hat{\mathbf{W}}_{t+1} &= \hat{\mathbf{W}}_{t+1}^t + \mathbf{K}_{t+1} (d_{t+1} - h(\hat{\mathbf{W}}_{t+1}^t, x_t)) \\ \mathbf{P}_{t+1}^{t+1} &= \mathbf{P}_{t+1}^t - \mathbf{K}_{t+1} \mathbf{j}_{t+1}^T \mathbf{P}_{t+1}^t \end{aligned} \quad (5)$$

The EKF is a suboptimal estimator based on linearization of the nonlinearity of the underlying Neural Network. It provides an approximation of the state mean \mathbf{W}_t and the state covariance \mathbf{P}_t . The matrix \mathbf{K}_t is the Kalman gain.

After computing the estimates \mathbf{W}_t and \mathbf{P}_t by equations [1] the Rauch-Tung-Striebel smoother [12] is utilised for recursively computing corrections to the EKF estimates. This is achieved through the following backward recursions:

$$\begin{aligned}\mathbf{S}_{t-1}^{t-1} &= \mathbf{P}_{t-1}^{t-1}(\mathbf{P}_t^{t-1})^{-1} \\ \hat{\mathbf{W}}_{t-1}^N &= \hat{\mathbf{W}}_{t-1}^{t-1} + \mathbf{S}_{t-1}^{t-1}(\hat{\mathbf{W}}_t^N - \hat{\mathbf{W}}_{t-1}^{t-1}) \\ \mathbf{P}_{t-1}^N &= \mathbf{P}_{t-1}^{t-1} + \mathbf{S}_{t-1}^{t-1}(\mathbf{P}_t^N - \mathbf{P}_t^{t-1})(\mathbf{S}_{t-1}^{t-1})^T \\ \mathbf{P}_{t,t-1}^N &= \mathbf{P}_t^t(\mathbf{S}_{t-1}^{t-1})^T + \mathbf{S}_t^t(\mathbf{P}_{t+1,t}^N - \mathbf{P}_t^{t-1})(\mathbf{S}_{t-1}^{t-1})^T\end{aligned}\tag{6}$$

The extended Kalman smoother provides a minimum variance Gaussian approximation to the posterior probability density function $p(\mathbf{W}|x_{1:N})$.

4 Expectation Maximisation Learning

The EM algorithm is an iterative method for finding a mode of the likelihood function $p(x_{1:N}|\mathbf{R}, \mathbf{Q})$. The algorithm alternates between two steps, the E-step (expectation) and the M-step (maximisation). In the E-step, an estimate of the state \mathbf{W} given the data $x_{1:N}$ and parameters $\boldsymbol{\theta} = [\mathbf{R}, \mathbf{Q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}]$ is produced, and in the M-step, the parameters $\boldsymbol{\theta}$ are then estimated given the new state.

Maximum likelihood estimation of the parameters \mathbf{W} are found through maximising the complete likelihood of the data, assuming Markovian state evolution and uncorrelated state and measurement noise:

$$p(\mathbf{W}, x_{1:N}|\boldsymbol{\theta}) = p(\mathbf{W}_1|\boldsymbol{\theta}) \prod_{t=2}^N p(\mathbf{W}_t|\mathbf{W}_{t-1}, \boldsymbol{\theta}) \prod_{t=1}^N p(d_t|\mathbf{W}_t, \boldsymbol{\theta})\tag{7}$$

It is assumed that the likelihood of the data given the states, and the evolution of the states is Gaussian.

$$p(\mathbf{W}_1|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{q/2}|\boldsymbol{\Sigma}|^{1/2}} \exp[-\frac{1}{2}(\mathbf{W}_1 - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{W}_1 - \boldsymbol{\mu})]\tag{8}$$

$$p(\mathbf{W}_t|\mathbf{W}_{t-1}, \boldsymbol{\theta}) = \frac{1}{(2\pi)^{q/2}|\mathbf{Q}|^{1/2}} \exp[-\frac{1}{2}(\mathbf{W}_t - \mathbf{W}_{t-1})\mathbf{Q}^{-1}(\mathbf{W}_t - \mathbf{W}_{t-1})]\tag{9}$$

$$p(x_t|\mathbf{W}_t, \boldsymbol{\theta}) = \frac{1}{(2\pi)^{m/2}|\mathbf{R}|^{1/2}} \exp[-\frac{1}{2}(d_t - h(\mathbf{W}_t, x_t))\mathbf{R}^{-1}(d_t - h(\mathbf{W}_t, x_t))]\tag{10}$$

By taking the log of the likelihood of the complete data we arrive at

$$\begin{aligned} \ln p(\mathbf{W}, x_{1:N} | \boldsymbol{\theta}) = & - \sum_{t=1}^N \left[\frac{1}{2} (d_t - h(\mathbf{W}_t, x_t)) R^{-1} (d_t - h(\mathbf{W}_t, x_t)) \right] - \frac{N}{2} \ln |R| \\ & - \sum_{t=2}^N \left[\frac{1}{2} (\mathbf{W}_t - \mathbf{W}_{t-1}) \mathbf{Q}^{-1} (\mathbf{W}_t - \mathbf{W}_{t-1}) \right] - \frac{N-1}{2} \ln |\mathbf{Q}| - \frac{1}{2} (\mathbf{W}_1 - \boldsymbol{\mu}) \boldsymbol{\Sigma}^{-1} (\mathbf{W}_1 - \boldsymbol{\mu}) \\ & - \frac{1}{2} \ln |\boldsymbol{\Sigma}| - \frac{N(m+q)}{2} \ln(2\pi) \end{aligned} \quad (11)$$

Taking the Expectation of both sides of the equation and differentiating the expected log-likelihood with respect to R^{-1}

$$\begin{aligned} \frac{\partial \mathbb{E}[\ln p(\mathbf{W}, x_{1:N} | \boldsymbol{\theta})]}{\partial R^{-1}} \approx & \frac{1}{2} \frac{\partial}{\partial R^{-1}} \left(\frac{N}{2} \ln |R^{-1}| - \sum_{t=1}^N \text{tr}(R^{-1} [\mathbf{j}^T \mathbf{P}_t^N \mathbf{j} \right. \\ & \left. + (d_t - h(\hat{\mathbf{W}}_t, x_t)) (d_t - h(\hat{\mathbf{W}}_t, x_t))^T]) \right) \\ = & \frac{N}{2} - \sum_{t=1}^N \frac{1}{2} (\mathbf{j}^T \mathbf{P}_t^N \mathbf{j} + (d_t - h(\hat{\mathbf{W}}_t, x_t)) (d_t - h(\hat{\mathbf{W}}_t, x_t))^T) \end{aligned} \quad (12)$$

and setting the resulting solution equal to zero and solving for R results in

$$R = \frac{1}{N} \sum_{t=1}^N (\mathbf{j}^T \mathbf{P}_t^N \mathbf{j} + (d_t - h(\hat{\mathbf{W}}_t, x_t)) (d_t - h(\hat{\mathbf{W}}_t, x_t))^T) \quad (13)$$

Similarly, differentiating with respect to \mathbf{Q}

$$\frac{\partial}{\partial \mathbf{Q}^{-1}} \mathbb{E}[\ln p(\mathbf{W}, x_{1:N} | \boldsymbol{\theta})] \approx \frac{N-1}{2} \mathbf{Q} - \frac{1}{2} (\mathbf{C} - 2\mathbf{B}^T + \mathbf{A}^T) \quad (14)$$

where \mathbf{Q} is a diagonal matrix. Equating to zero and solving for \mathbf{Q} leads to

$$\mathbf{Q} = \frac{1}{N-1} (\mathbf{C} - 2\mathbf{B}^T + \mathbf{A}^T) \quad (15)$$

where we define the quantities

$$\begin{aligned} \mathbf{A} &= \sum_{t=1}^N (\mathbf{P}_{t+1}^N + \hat{\mathbf{W}}_{t-1}^N (\hat{\mathbf{W}}_{t-1}^N)^T) \\ \mathbf{B} &= \sum_{t=1}^N (\mathbf{P}_{t,t+1}^N + \hat{\mathbf{W}}_t^N (\hat{\mathbf{W}}_{t-1}^N)^T) \\ \mathbf{C} &= \sum_{t=1}^N (\mathbf{P}_t^N + \hat{\mathbf{W}}_t^N (\hat{\mathbf{W}}_t^N)^T) \end{aligned} \quad (16)$$

It is also possible to solve for the initial conditions in the M step, via taking the derivative of the expected log-likelihood with respect to the initial mean

$$\frac{\partial}{\partial \mu} \mathbb{E}[\ln p(\mathbf{W}, x_{1:N} | \boldsymbol{\theta})] \approx \frac{1}{2} \boldsymbol{\Sigma}^{-1} (-2\hat{\mathbf{W}}_1^N + 2\boldsymbol{\mu}) \quad (17)$$

which leads to the initial value of $\boldsymbol{\mu} = \mathbf{W}_1^N$ and similarly for the covariance, taking the derivative with respect to $\boldsymbol{\Sigma}$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}^{-1}} \mathbb{E}[\ln p(\mathbf{W}, x_{1:N} | \boldsymbol{\theta})] \approx \frac{1}{2} \boldsymbol{\Sigma} - \frac{1}{2} (\hat{\mathbf{W}}_1^N - \boldsymbol{\mu})(\hat{\mathbf{W}}_1^N - \boldsymbol{\mu})^T + \mathbf{P}_1^N \quad (18)$$

leads to the initial covariance $\boldsymbol{\Sigma} = \mathbf{P}_1^N$.

4.1 The EM Steps

The algorithm starts off with an initial guess for $\boldsymbol{\theta}$. Then, in the E-step, the expected values of $\hat{\mathbf{W}}_t^N$, \mathbf{P}_t^N and $\mathbf{P}_{t,t-1}^N$ are obtained from their current estimates through extended Kalman filtering and smoothing. In the M-step, the new values of $\boldsymbol{\theta}$ are obtained using the above equations.

5 Experimental Results

To assess the performance of the proposed model, we have implemented several training algorithms that are similar to the presented EM-RNN, including the RNN-RTRL algorithm [18], and the Extended Kalman Filter trained recurrent network (RNN-EKF). Here we consider hourly observations of the D_{st} index, along with observations of the b_x , b_y , and b_z components of the IMF index. These exogenous variables were selected due to previous research that has indicated that sensors measuring IMF components are less prone to error during periods of high geomagnetic activity [10]. The data considered in this study consisted of 9000 data points ranging from January 1, 1980 to February 5, 1981. Three studies were conducted to assess the model's ability to perform during periods of geomagnetic instability (storms), which correspond to the dates of March 31 to August 14 1980 for the mild series of storms, and from December 12, 1980 to December 24, 1980 for the severe storm. The quiet period ranged from December 25, 1980 to February 5, 1981. In building the model, we used the period ranging from January 1 1980 to March 31, 1980.

The forecast errors were measured by the root mean squared error (RMSE) computed by $RMSE = ((1/T) \sum_{t=1}^T (d_t - y_t)^2)^{1/2}$, where T is the length of the data set. In all simulations, the weights of the networks were initialised with random uniformly distributed weights in the range of $[-2, 2]$. Each of the recurrent networks were initialised with 3 hidden neurons and 3 input neurons for each factor (b_z , b^2 , b_y^2 , D_{st}), resulting in a total input window of size 12. All RNNs had one output neuron corresponding to the one hour ahead value of the D_{st} signal. For the EKF trained network, the initial diagonal elements of the covariance matrix of the $[\mathbf{Q}]_{ii}$ and R were set to $1.0e^{-3}$ and $1.0e^{-2}$ respectively, and for the EM trained models, the diagonal of the matrix $[\mathbf{Q}]_{ii}$ was set to .01 and R was set to 10. The learning rate for the RTRL algorithm was set to .05.

Plots of one Hour Ahead Forecasts of the D_{st} Index

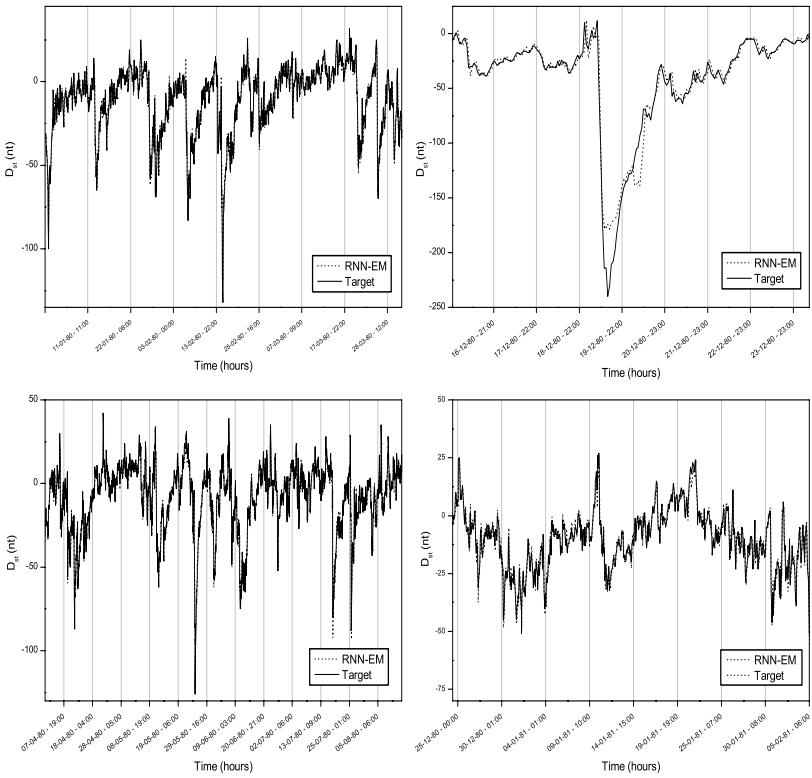


Fig. 2. Plots of the EM-RNN and D_{st} measurements. The top left plot is of the in sample fitting of model to the data. The lower left plot shows out of sample forecast performance on a series of mild storms. The lower right hand plot shows the model performance during a quiet period (no storm), and in the top right plot, the model performance on the severe storm is provided.

5.1 Forecasting D_{st}

Table II summarises the experimental results of single step ahead model fitting and prediction of the storms. The in sample performance of EM and the EKF were quite similar, and the gradient descent trained RNN had the worst fit. In both the severe and mild storm forecasting tasks, the EM-RNN outperforms the other recurrent network training algorithms including the recurrent network trained by RTRL and the EKF. However, in the quiet period, the EM-RNN and the RNN-EKF perform similarly. The plots of the experiments are given in Figure 2. The use of the EM algorithm has led to improvements in out of sample predictions over the EKF, however, the EM-RNN more closely approximates the target series than the remaining algorithms as shown in the given plots. As shown in the plot of the severe storm, the forecast model is unable to capture the entire amplitude of the storm. This is most likely due to the training set which does not contain

Table 1. Numerical Performance of the Studied Algorithms on One Hour Ahead Forecasts of the D_{st} index

Model	RMSE(<i>training</i>)	RMSE(<i>severe</i>)	RMSE(<i>mild</i>)	RMSE(<i>quiet</i>)
RNN-RTRL	9.64	21.17	11.83	12.78
RNN-EKF	3.57	16.62	4.04	3.16
RNN-EM	3.41	11.78	3.65	3.24

any severe storms (i.e. the maximum negative amplitude is $-150nT$ which is similar to what the model predicts in the case of the severe storm. Further studies will investigate predictions with different data sets, some which include severe negative geomagnetic activity.

6 Conclusion

RNNs have been found to be useful in modeling space weather phenomena. Most of the literature in neural space weather forecasting has focused on finding relevant inputs to the RNN, and little effort has been spent on improving the recurrent model itself. This paper presented a probabilistic training algorithm for dynamic recurrent neural models, which has improved D_{st} index forecasts over the current state of the art reported in the literature. Through experimental simulations, the paper has demonstrated that the EM training of the RNN is able to improve forecasts of D_{st} for one hour ahead forecasts. The presented results are encouraging as they show that the EM-RNN has the capacity to accurately model complex Geo-magnetic phenomena.

References

1. Axford, W.I., Hines, C.O.: A unifying theory of high-latitude geophysical phenomena and geomagnetic storms. *Can. J. Phys.* 39, 1433–1464 (1961)
2. de Freitas, J.F.G., Niranjan, M., Gee, A.H.: Dynamic Learning with the EM Algorithm for Neural Networks. *J. Vlsi. Signal. Proc.* 26, 119–131 (2000)
3. Dungey, J.W.: Interplanetary magnetic field and the auroral zones. *Phys. Rev. Lett.* 26, 47–48 (2000)
4. Freeman, J., Natal, A., Reiff, P., Denig, W., Gussenhoven-Shea, S., Heinemann, M., Rich, F., Hairston, M.: The use of neural networks to predict magnetospheric parameters for input to a magnetospheric forecast model. In: Proceedings of Artificial Intelligence Applications in Solar-Terrestrial Physics Workshop, pp. 167–181 (1993)
5. Gonzales, W.D., Joselyn, J.A., Kamide, Y., Kroehl, H.W., Rostoker, G., Tsurutani, B.T., Vasiliunas, V.M.: What is a geomagnetic storm? *J. Geophys. Res.* 99, 5771–5792 (1994)
6. Ghahramani, Z., Hinton, G.E.: Parameter estimation for linear dynamical systems. Technical Report, University of Toronto CRG-TR-96-2 (1996)
7. Lundstedt, H.: Neural Networks and prediction of solar-terrestrial effects. *Planet. Space Sci.* 40, 457–464 (1992)
8. Lundstedt, H., Wintoft, P.: Prediction of geomagnetic storms from solar wind data with the use of a neural network. *Ann Geophys.* 12, 19–24 (1994)
9. Lundstedt, H., Gleisner, H., Wintoft, P.: Operational forecasts of the geomagnetic Dst index. *Geophys. Res. Lett.* 29(24), 34–1–34–4 (2002)

10. Pallocchia, G., Amata, E., Consolini, G., Marcucci, M.F., Bertello, I.: Geomagnetic Dst index forecast based on IMF data only. *Ann. Geophys.* 24, 989–999 (2006)
11. Puskorius, G.V., Feldkamp, L.A.: Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. *IEEE T Neural Networks* 5(2), 279–297 (1994)
12. Rauch, H.E., Tung, F., Striebel, C.T.: Maximum Likelihood Estimates of Linear Dynamic Models. *AIAA Journal* 3(8), 1445–1450 (1965)
13. Schottky, B., Saad, D.: Statistical mechanics of EKF learning in neural networks. *J. Phys. A* 32(9), 1605–1621 (1999)
14. Shumway, R.H., Stoffer, D.S.: An Approach To Time Series Smoothing and Forecasting Using the EM Algorithm. *J. Time Ser. Anal.* 4, 253–264 (1982)
15. Wei, H.L., Bilings, S.A., Balikhin, M.A.: Prediction of the Dst index using multiresolution wavelet models. *J. Geophys. Res.* 109(A7), A07212 (2004)
16. W.H.L., Bilings, S.A., Balikhin, M.A.: Forecasting the Geomagnetic Activity of teh Dst index using Multiscale Radial Basis Function Networks. In: *Advances in Space Research*, vol. 40, pp. 1863–1870 (2007)
17. Wu, J.G., Lundstedt, H.: Geomagnetic storm predictions from solar wind data with the use of dynamic neural networks. *J. Geophys. Res.* 102(A7), 14–255 (1997)
18. Williams, R.J., Zipser, D.: A learning algorithm for continuously running fully connected recurrent neural networks. *Neural Computation* 1, 270–280 (1989)

On the Quantification of Dynamics in Reservoir Computing

David Verstraeten* and Benjamin Schrauwen

Department of Electronics and Information Systems (ELIS), Ghent University

Sint Pietersnieuwstraat 41, 9000 Ghent, Belgium

David.Verstraeten@UGent.be

Abstract. Reservoir Computing (RC) offers a computationally efficient and well performing technique for using the temporal processing power of Recurrent Neural Networks (RNNs), while avoiding the traditional long training times and stability problems. The method is both simple and elegant: a random RNN (called the *reservoir*) is constructed using only a few global parameters to tune the dynamics into a desirable regime, and the dynamic response of the reservoir is used to train a simple linear regression function called the readout function - the reservoir itself remains untrained. This technique has shown some experimentally very convincing results on a variety of tasks, but a thorough understanding of the importance of the dynamics for the performance is still lacking. This contribution aims to extend this understanding, by presenting a more sophisticated extension on the traditional way of characterizing the reservoir dynamics, by using the dynamic profile of the Jacobian of the reservoir instead of static, *a priori* measures such as the standard spectral radius. We show that this measure gives a more accurate description of the reservoir dynamics, and can serve as predictor for the performance. Additionally, due to the theoretical background from dynamical systems theory, this measure offers some insight into the underlying mechanisms of RC.

1 Introduction: Reservoir Computing

Reservoir Computing (RC) [15], an idea that was originally independently introduced as Echo State Networks (ESN) [5] and Liquid State Machines (LSM) [7] has grown in the last few years into a research subfield that has attracted quite some attention. This is likely due to the attractive properties of the method: it can be used to solve temporal learning tasks without extensive parameter tuning or long training times and is easy to use and understand.

Reservoir Computing relies on the dynamic response of an excitable, (usually) nonlinear medium - the *reservoir* - to a one- or multidimensional input signal. The state of the system - which is in effect a nonlinear transformation with fading memory of the input - is then used as input for a linear regression function, which can be trained using any of the available online or offline training methods for

* Corresponding author.

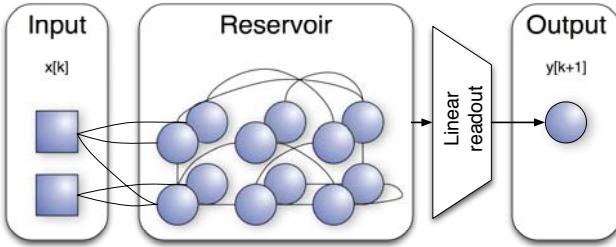


Fig. 1. Schematic view of an RC system, with the input signals (left) driving the reservoir, which is then used as input for the linear readout that in turn extracts the output

linear classifiers or regressors (see Fig. 1). In a sense, the functionality of the reservoir can be interpreted as being a random, temporal and nonlinear kernel [11], which performs a temporal-to-spatial transformation. This transformation boosts the computational capabilities of the subsequent linear classifier, enabling it to solve problems it would not be able to without the reservoir. Thus, in RC, two difficult tasks needed for solving temporal classification or regression problems are elegantly separated, namely the nonlinear dynamic preprocessing, and the training of the actual classifier.

In practice, the methodology to construct and train an ESN system can be summarized as follows:

- Construct a recurrent neural network consisting of N nodes with sigmoid (\tanh) nonlinearities. The weights of the $N \times N$ reservoir weight matrix \mathbf{W}_{res} and an $M \times N$ input weight matrix \mathbf{W}_{in} are drawn from a random distribution (e.g. a gaussian distribution), or a from discrete set. Rescale \mathbf{W}_{res} globally, such that the spectral radius of \mathbf{W}_{res} is set to the desired value. The spectral radius of a matrix is its largest absolute eigenvalue, and denoted as $\rho(\mathbf{W}_{\text{res}})$. The rationale behind the rescaling is explained later. Rescale \mathbf{W}_{in} with a constant value, the *input scale factor* (usually around 1).
- Simulate the network by driving it with an external (possibly multidimensional) input signal $\mathbf{u}[k]$. The network state at time k is denoted as $\mathbf{s}[k]$. The network is simulated recursively, in a timestep based way, as follows: $\mathbf{s}[k+1] = f(\mathbf{W}_{\text{res}}\mathbf{s}[k] + \mathbf{W}_{\text{in}}\mathbf{u}[k])$.
- Compute the output weights by least squares regression on the matrix \mathbf{A} - which is a concatenation of all vectors $\mathbf{s}[k]$ - using the desired output matrix \mathbf{o} as the right-hand side. I.e., compute the matrix \mathbf{W}_{out} that satisfies the following equation: $\mathbf{W}_{\text{out}} = \min_{\mathbf{W}} \|\mathbf{AW} - \mathbf{o}\|^2$. In practice, this can be done in a single step by using the Moore-Penrose generalized matrix inverse [10], or *pseudo-inverse* \mathbf{A}^\dagger of the matrix \mathbf{A} , which is defined as : $\mathbf{A}^\dagger = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$, as follows: $\mathbf{W}_{\text{out}} = \mathbf{A}^\dagger \mathbf{o} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{o}$.

- Simulate the network on the test set in the same way as above, and compute the output as follows: $\hat{\mathbf{o}}[k] = \mathbf{W}_{\text{outs}}[\mathbf{s}[k]]$.
- Evaluate the performance based on the difference between the RC output $\hat{\mathbf{o}}[k]$ and the target output $\mathbf{o}[k]$.

While RC as a research area is rooted in neural network research and is still mainly active there, its ideas are extendable to other fields. The basic idea of using a nonlinear dynamic system to act as a complex preprocessing filter for the linear readout is very powerful, and can be ported to other research areas. More exotic incarnations of RC have already been described in literature, which include (ranked in order of ‘deviation from the standard RNN’) : bandpass reservoirs [12,18], Cellular Nonlinear Network (CNN) reservoirs [16] and a reservoir built from coupled nano-photonic nonlinear components [13].

2 Disadvantages of Static Reservoir Measures

As was mentioned in the brief description of the ESN methodology, the spectral radius is an important parameter that controls the dynamic regime of the reservoir. It amounts to a global scaling of the eigenvalues of the connection matrix. From a system theoretic point of view, this can be interpreted as follows: for a small-signal approximation (i.e. the state of the reservoir remains near the zero fix-point), the reservoir can be approximated as a linear time-invariant, discrete-time system:

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{Ax}[k] + \mathbf{Bu}[k] \\ \mathbf{y}[k+1] &= \mathbf{Cx}[k+1] + \mathbf{Du}[k+1]\end{aligned}$$

where $\mathbf{x}[k]$ represents the state of the reservoir (the vector of neuron activations) at time k , and $\mathbf{u}[k]$ and $\mathbf{y}[k]$ represent the input and output to the system, respectively. The matrix \mathbf{A} contains the internal weights of the reservoir (\mathbf{W}_{res} from above), the \mathbf{B} matrix contains the input-to-reservoir weights (\mathbf{W}_{in} from above), and \mathbf{C} and \mathbf{D} contain the (trained) reservoir-to-output (\mathbf{W}_{out} from above) and input-to-output weights respectively (the latter is usually left zero).

It follows from linear system theory [2] that if the matrix A has all singular values smaller than 1^{I} , it is definitely stable, while if any absolute eigenvalue (i.e. spectral radius) is larger than 1, the system (i.e. the reservoir) will surely be unstable in the sense that it will deviate unboundedly from the fixed point when started from a non-zero state. However, the reservoirs of the ESN type (and the reservoirs that we will consider in this contribution), have a squashing $\tanh()$ nonlinearity, that counteracts this unbounded growth - which means that the norm of the state vector of the reservoir will always remain bounded. This nonlinearity also means that the spectral radius as a stability measure loses its significance when the system deviates from an ϵ -region around the zero state.

¹ This implies that the maximal gain in any direction in state space is smaller than one, and the system is always contracting.

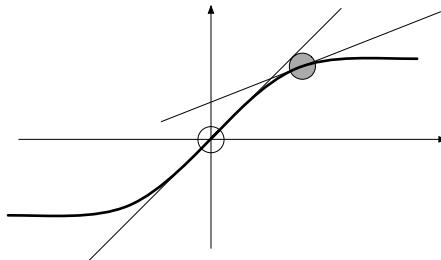


Fig. 2. The gain of the sigmoid nonlinearity is largest around the origin (white circle). Once the neuron is driven by an external signal or a constant bias, the working point shifts up or downward (gray circle) and the local gain decreases, resulting in a less dynamically excitable reservoir.

This means that it is possible that reservoirs with a spectral radius larger than one do possess the echo state property when driven by an external input - this was e.g. proven experimentally in [14]. Once the system is driven by an external input or if a constant bias is fed to the nodes, the operating point of all neurons shifts along the nonlinearity and the effective local gain (i.e. the slope of the tangent in the operating point) becomes smaller (see Fig. 2).

In [5], the linear approximation described above is used to derive some mathematically founded guidelines for constructing weight matrices for suitable reservoirs. The suitability of a reservoir in this case is mainly identified through the presence of the so-called *echo state property*, which roughly states that the state of the reservoir is only determined by the inputs from a sufficiently long time in the past, and that the initial state of the reservoir eventually gets washed out. Several bounds for this property have been described in literature:

- A reservoir whose weight matrix \mathbf{W}_{res} has a largest singular value (LSV) (denoted as $\bar{\sigma}(\mathbf{W}_{res})$) smaller than one, is guaranteed to have the echo state property. However, in practice this guideline is of little use since these reservoirs are not dynamically rich enough to perform well.
- A reservoir whose weight matrix has a spectral radius (SR) - i.e. a largest absolute eigenvalue - larger than one is guaranteed *not* to have the echo state property. So, $\rho(\mathbf{W}_{res}) \leq 1$ is a necessary condition for the echo state property. While the spectral radius criterium is not a sufficient condition, in practice it is used as a guideline for constructing good reservoirs for many problems.
- In [1], a tighter bound on the echo state property than $\bar{\sigma}(\mathbf{W}_{res}) < 1$ was presented. A Euclidean weighted matrix norm $\|\mathbf{W}\|_{\mathbf{D}} = \|\mathbf{DWD}^{-1}\|_2 = \bar{\sigma}(\mathbf{DWD}^{-1})$ was introduced, and it turns out that for a certain class of structured weighting matrices \mathbf{D}_δ , the relation $\rho(\mathbf{W}) < \inf_{\mathbf{D}_\delta} \|\mathbf{D}_\delta \mathbf{WD}_\delta^{-1}\| < \bar{\sigma}(\mathbf{W})$ holds. The center term $\inf_{\mathbf{D}_\delta} \|\mathbf{D}_\delta \mathbf{WD}_\delta^{-1}\|$ is called the structured singular value μ_{SSV} , a quantity widely used in robust control theory. It turns out that μ_{SSV} offers a bound on the echo state property (namely, $\mu_{SSV} < 1$)

that is less conservative than the standard $\overline{\sigma}(\mathbf{W}_{res}) < 1$. However, while this new bound is an improvement over the standard LSV and SR bounds, it is computationally quite demanding to evaluate (21 seconds for a reservoir of 500 nodes, versus .6 seconds to compute the spectral radius).

All the quantities described above are static measures that only take the internal reservoir weight matrix into account and disregard other factors such as input scaling, bias or dynamic range of the input signals - factors that are equally important in defining the dynamic properties of the system. Clearly, an accurate way of quantifying the dynamics of the reservoir, evaluated in the current working point of the reservoir, would be very useful. This notion is explored further in the next section.

3 Quantifying Dynamic Properties of Reservoirs

The readout of the RC system is quite unsophisticated in terms of computational power: it is both linear and memoryless. While these properties enable the application of easy and optimal training algorithms, this also means that complex temporal problems cannot be solved by the linear readout alone. Thus, the functionality of the reservoir is twofold: it should perform a suitably nonlinear transformation of the input so that the discriminating power of the linear readout gets boosted, and it should also offer a fading memory of past inputs to the readout. In some ways, these two functions are contrary to each other: it was shown theoretically that linear networks and even long delay lines have the largest memory of past inputs [4][7][3], but many non-trivial tasks require at least some form of nonlinear behaviour, which reduces the memory of the network. Thus, a good reservoir should ideally find the optimal trade-off between these two opposing goals. This is closely linked to the dynamic regime of the reservoir.

The dynamic properties of the reservoir at a given point in time are determined by a couple of factors: the reservoir weight matrix (this was discussed in the previous section), an optional bias, the nonlinearity of the nodes and the external input that drives the reservoir. These factors determine the operating regime of the reservoir, and as such the local gain of the system at any given time. Here, we will discuss two (related) tools for quantifying these dynamic properties, namely the local Lyapunov exponent (LLE) and the Jacobian of the reservoir. The Jacobian \mathbf{J}_f of a map $f(s)$ is given by:

$$\mathbf{J}_f(\mathbf{s}) = \begin{pmatrix} \frac{\partial f_1}{\partial s_1}(\mathbf{s}) & \dots & \frac{\partial f_1}{\partial s_n}(\mathbf{s}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial s_1}(\mathbf{s}) & \dots & \frac{\partial f_n}{\partial s_n}(\mathbf{s}) \end{pmatrix},$$

where $s = [s_1 s_2 \dots s_n]$ is the vector of activation values of the neurons in the reservoir, and f is the nonlinearity of the nodes - in this case a $\tanh()$. The matrix contains the local derivative of every state value w.r.t. every other state value. In the case of a $\tanh()$ reservoir, this simplifies to:

$$\mathbf{J}_f(\mathbf{s}) = \text{diag}[1 - s_1^2[k], 1 - s_2^2[k], \dots, 1 - s_n^2[k]] \mathbf{W}_{res},$$

where the notation $\text{diag}[\cdot]$ signifies a diagonal matrix with the given values on the diagonal, and \mathbf{W} is the weight matrix of the reservoir. From this, the k th LLE $\tilde{\lambda}_k$ can then be approximated as: $\tilde{\lambda}_k = \log\left(\prod_{n=1}^N(r_k)^{1/n}\right)$ with N the number of timesteps in the trajectory that is considered, and r_k the k th eigenvalue of \mathbf{J}_f . The LLE offers a local estimation of the predictability or excitability of a dynamic system around a certain point in state space. It is only an approximation of the true Lyapunov spectrum for two reasons: first of all, we only consider a finite trajectory while the definition of the Lyapunov exponent requires that $N \rightarrow \infty$, and secondly the system under consideration is driven by an external input signal. However, we argue that this spectrum can still offer a valid quantification of the local dynamic properties of the reservoir.

In [15], the relationship between the mean of the maximum of the local Lyapunov spectrum and the performance of the reservoir was studied, and it was found that for a given task, the optimal performance of a reservoir was consistently attained for the same value of the maximal LLE. While this finding was useful from a theoretical point of view because it offered a more refined measure of the reservoir dynamics than the stationary measures mentioned in the previous section, it does not supply a practical means for choosing the reservoir dynamics or offers insight into the meaning of this metric.

Closer inspection of the *complete* local Lyapunov spectrum reveals another, and in some ways more useful phenomenon. Figure 3 shows a plot of the mean over time of all LLEs as the spectral radius of the reservoir is varied from .1 to 3 and the reservoir is driven by noise (which is the input for the NARMA task, see below). The plot shows that the maximal exponent increases monotonically (as was shown previously in [15]), but also that the *minimal* exponent reaches a maximum for a spectral radius of 1, and then decreases again. Thus, the bundle of LLEs becomes narrower and then broader again as the spectral radius of the reservoir weight matrix is increased. More importantly, the maximum of the minimal lyapunov exponent is a good predictor for the optimal performance of the system. In the next section, we will present some more elaborate experimental results and discuss the implications of this phenomenon.

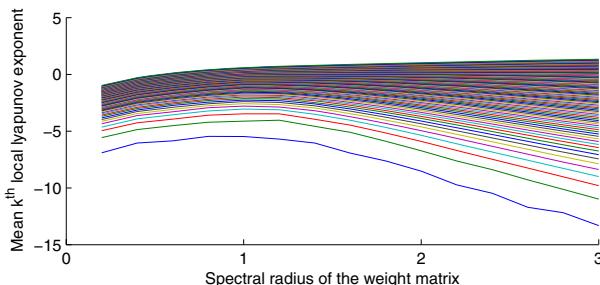


Fig. 3. The full mean (over time) local lyapunov spectrum for a reservoir of 100 nodes for the NARMA task

4 Experiments and Discussion

The maximal LE is - for autonomous systems - an indicator of chaotic behaviour: if it is larger than zero the system is said to be chaotic, meaning that perturbations from a trajectory are amplified exponentially in at least one direction. At first sight no such interpretation exists for the minimal LE - it simply quantifies the direction of minimal expansion of the system. However, closer inspection reveals that a more informative interpretation is possible by inspecting the Jacobian matrix itself.

We start with the following remark: when evaluating the Jacobian around the origin in state space (zero fixpoint, i.e. $s = 0$), it reduces to the weight matrix \mathbf{W} of the reservoir, and its largest eigenvalue is precisely the spectral radius of the reservoir. Therefore, the eigenvalue spectrum of the Jacobian can be seen as a dynamic extension of the static eigenvalue spectrum of the weight matrix (which was the subject of previous work on dynamics in reservoirs, e.g. [9]). Moreover, the lyapunov spectrum at a single point in time, given by $\log(\text{eig}(\mathbf{J}_f^T \mathbf{J}_f))$, is equal to the log of the singular value spectrum of the Jacobian itself². Following this line of reasoning, we measured the minimal singular value (SV) of the Jacobian³ and computed its mean over time as we vary the spectral radius of the reservoir weight matrix, and the scaling factor of the input matrix. We then compared this measure with the performance on two tasks:

- The Mackey-Glass timeseries prediction. This mildly chaotic timeseries (with delay parameter $\tau = 17$) is a common benchmark and RC systems have shown very good performance on this task [6]. The RC system was trained to do one-step ahead prediction on a training timeseries of 4000 timesteps, and was then used to autonomously generate the signal by feeding its own prediction back as input into the reservoir. The performance is evaluated as the first timestep when the divergence (expressed as the absolute error) between the predicted and target signal exceeds 0.1.
- Modelling a 30th order Nonlinear AutoRegressive Moving Average (NARMA) system. Here, the input $u[k]$ to the network is a random signal sampled from a uniform distribution in $[0, .5]$, and the target output is given by $y[k+1] = 0.2y[k] + 0.04y[k](\sum_{i=0}^{29} y[k-i]) + 1.5u[k-29]u[k] + 0.001$. The performance is measured with the normalized root mean square error (NRMSE).

Figure 4 shows the mean maximal LLE, the mean minimal singular value of the Jacobian, and the score on both tasks, as the spectral radius and scaling factor of the input matrix are swept within the plausible range $[.1, 2]$ with steps of .1 (every point in the plots represents the average over twenty different reservoir instantiations). The top plots show the same measure that was introduced in [15]. This measure clearly does not capture all necessary dynamic properties of the

² In general, for a matrix \mathbf{M} , the squares of its singular values are equal to the eigenvalues of $\mathbf{M}^T \mathbf{M}$.

³ At every 50th timestep for computational reasons, but this provides sufficient accuracy.

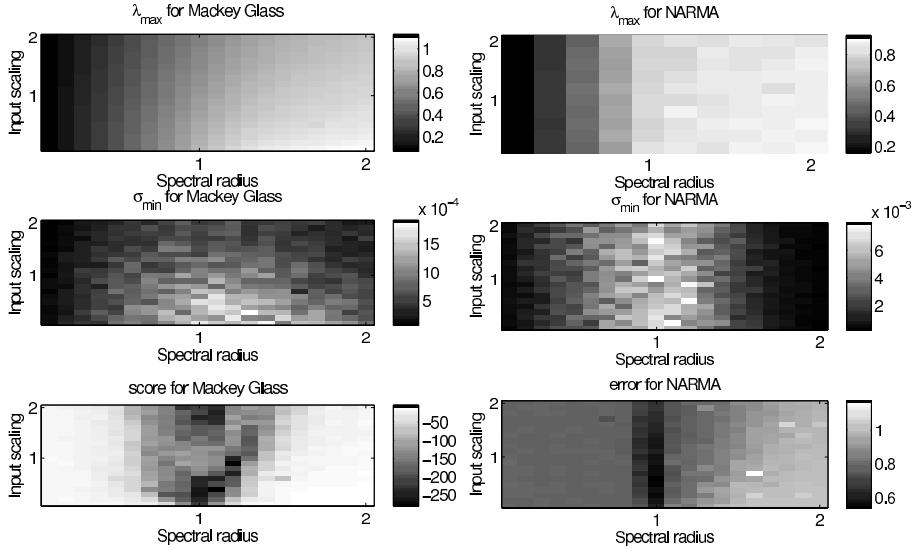


Fig. 4. The top plots show the maximal LLE, the middle plots show the minimal SV and the bottom plots show the performance for the Mackey-Glass prediction (left) and NARMA (right) task. Note that for the Mackey-Glass performance plot, higher is better while for NARMA lower is better.

reservoir, since it increases monotonically with the spectral radius, and the input scaling has hardly any influence. The middle plots on the other hand - which show the minimal SV σ_m - offer a much more nuanced image. The minimal SV σ_m varies with both the spectral radius and the input scaling - which indicates that it captures the changing dynamical properties of the reservoir as a function of the scaling parameters quite well. Moreover, the area of optimal performance (bottom plots) coincides quite nicely with the areas where σ_m is highest. Thus, σ_m is a more accurate predictor of performance than both the largest LLE and the spectral radius.

The interpretation of σ_m of the Jacobian is at first sight not trivial: it simply qualifies the minimal gain of the system in any direction in state space. However, σ_m can be written as the ratio between the norm $\|\mathbf{J}_f\|$ and the condition number $\kappa(\mathbf{J}_f)$ of the Jacobian: $\sigma_m = \frac{\|\mathbf{J}_f\|_2}{\kappa(\mathbf{J}_f)}$, since $\sigma_m^{-1} = \|\mathbf{J}_f^{-1}\|$ and $\kappa(\mathbf{J}_f) = \|\mathbf{J}_f^{-1}\| \|\mathbf{J}_f\|$ and where $\|\cdot\|_2$ denotes the l_2 norm.

This relation yields an interesting interpretation. In the field of robotics (which borrows substantially from dynamical system theory), both the condition number and the norm of the Jacobian are widely used measures for quantifying the dynamic behaviour of e.g. robotic manipulators [8]. In particular, the norm of the Jacobian is a measure of the maximal gain of the system in any direction, while the condition number is used to quantify the dexterity of the robot arm or the closeness to a singular position (where the robot loses one or more degrees

of freedom due to constraints on the joints) - large condition numbers are an indication of low dexterity. When we transpose this interpretation to the reservoir, we can see that the maximization of σ_m is in fact a joint optimization of:

- the maximal gain of the system, thus ensuring good excitability and separation of the input signals in state space, and
- minimization of the condition number, which means that the dynamical system is far from singularity and has many degrees of freedom.

These two quantities are in opposition: if the gain of the reservoir is too high, the nodes will start to saturate and the expressive power of the nonlinearity decreases, which means that the reservoir is constrained to a lower-dimensional subspace of the state space. If it is too low, the reservoir does not separate the input signals enough. This trade-off is clearly present in the measure presented here.

One disadvantage of this measure is that it does not apply to linear reservoirs

- a maximization of the minimal SV of the jacobian (which is then just the reservoir weight matrix) results in unbounded weights.

5 Conclusions

While RC often achieves impressive performance on many tasks, the tuning of the parameters that control the dynamics of the reservoir is still a matter of expertise and manual experimentation, which is partly due to a lack of measures for accurately quantifying the dynamics of the reservoir. We have presented a novel metric for measuring the dynamical properties of the reservoir, and have shown that it is a more accurate predictor of performance than previously published measures. Moreover, we have given an interpretation of the measure that offers more insight into the functionality of the reservoir, showing that a trade-off is made between the excitability of the reservoir and its ‘degrees of freedom’ in state space.

Reservoir Computing has originated in the field of neural networks, but has since been extended to other, more generic implementations. For these more exotic reservoirs (such as reservoirs built from nano-photonics components [13]) especially, the standard tuning parameters such as spectral radius become meaningless. The measure introduced in this contribution can fill this void and offers a useful method for tuning and quantifying the dynamics of these novel reservoir implementations.

References

1. Buehner, M., Young, P.: A tighter bound for the echo state property. *IEEE Transactions on Neural Networks* 17(3), 820–824 (2006)
2. Csaki, F.: State-space methods for control systems. Akademiai Kiado (1977)
3. Ganguli, S., Huh, D., Sompolinsky, H.: Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences* 105(48), 18970 (2008)

4. Jaeger, H.: Short term memory in echo state networks. Technical Report GMD Report 152, German National Research Center for Information Technology (2001)
5. Jaeger, H.: Adaptive nonlinear system identification with echo state networks. In: Advances in Neural Information Processing Systems, pp. 593–600 (2003)
6. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. *Science* 308, 78–80 (2004)
7. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14(11), 2531–2560 (2002)
8. Merlet, J.P.: Jacobian, manipulability, condition number, and accuracy of parallel robots. *Journal of Mechanical Design* 128, 199 (2006)
9. Ozturk, M.C., Xu, D., Principe, J.C.: Analysis and design of echo state networks. *Neural Computation* 19, 111–138 (2006)
10. Penrose, R.: A generalized inverse for matrices, vol. 51(1955), pp. 406–413 (1955)
11. Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press, Cambridge (2004)
12. Siewert, U., Wustlich, W.: Echo-state networks with band-pass neurons: Towards generic time-scale-independent reservoir structures. Preliminary Report (October 2007)
13. Vandoorne, K., Dierckx, W., Schrauwen, B., Verstraeten, D., Baets, R., Bienstman, P., Campenhout, J.V.: Toward optical signal processing using photonic reservoir computing. *Optics Express* 16(15), 11182–11192 (2008)
14. Verstraeten, D., Schrauwen, B., Stroobandt, D.: Reservoir-based techniques for speech recognition. In: Proceedings of the World Conference on Computational Intelligence, pp. 1050–1053 (2006)
15. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: A unifying comparison of reservoir computing methods. *Neural Networks* 20, 391–403 (2007)
16. Verstraeten, D., Souza, S.X.-d., Schrauwen, B., Suykens, J., Stroobandt, D., Vandewalle, J.: Pattern classification with CNNs as reservoirs. In: Proceedings of the International Symposium on Nonlinear Theory and its Applications (NOLTA), vol. 9 (2008)
17. White, O.L., Lee, D.D., Sompolinsky, H.: Short-term memory in orthogonal neural networks. *Neural Comput. Phys. Rev. Lett.* 92, 148102 (2002)
18. Wyffels, F., Schrauwen, B., Verstraeten, D., Stroobandt, D.: Band-pass reservoir computing. In: Hou, Z., Zhang, N. (eds.) *Proceedings of the International Joint Conference on Neural Networks*, Hong Kong, June, pp. 3203–3208 (2008)

Solving the CLM Problem by Discrete-Time Linear Threshold Recurrent Neural Networks

Lei Zhang¹, Pheng Ann Heng^{1,2}, and Zhang Yi³

¹ Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

² School of Computer Science and Engineering, University of Electronic Science and Technology, Chengdu 610054, P.R. China

³ Machine Intelligence Laboratory, School of Computer Science, Sichuan University, Chengdu 610065, P.R. China

Abstract. The competitive layer model (CLM) can be described by the optimization problem that is formulated with the CLM energy function. The minimum points of CLM energy function can be achieved by running some proper recurrent neural networks. In other words, the CLM can be implemented by the recurrent neural networks. This paper proposes the discrete-time linear threshold recurrent networks to solve the CLM problem. The conditions for the stable attractors of the networks are obtained, which just correspond to the conditions of the minimum points of CLM energy function established in the literature before. Therefore, the proposed network can be used to implement the CLM.

1 Introduction

The Competitive Layer Model (CLM) was first proposed by Ritter in [1] as a model for spatial feature binding that may provide one of the basic sensory information processing principles [2]. In the mathematical point of view, CLM can be described by the optimization problem which is further formulated by an energy function, called CLM energy function, in some subspace of nonnegative orthant. In order to solve the CLM problem, it requires to find out the minimum points of the CLM energy function. Since the CLM is a model of recurrently connected structure, some recurrent neural networks (RNNs) can be used to solve CLM problem, i.e., implement CLM with RNNs by making the attractors of the RNNs correspond to the minimum points of the CLM energy function.

In [3], the continuous-time Linear-Threshold (LT) RNNs were used to implement the CLM for feature binding and sensory segmentation. Moreover, the foundations of implementing the CLM by Lotka-Volterra RNNs were totally established in [4]. However, the implementations of CLM are all based on the continuous-time RNNs so far. As we know, the discrete-time networks possess the advantages for direct computer simulations over digital simulation of the continuous-time neural network model. Since LT RNNs possess many good dynamical properties (see for example, [5,6]), while the dynamics of discrete-time

networks are generally different from the corresponding continuous-time ones [7], this paper proposes the discrete-time LT RNNs enabling to solve CLM problem.

The rest of the paper is organized as follows. In Section 2 a brief description of the CLM is presented. The model of discrete-time LT RNNs used to solve the CLM problem is proposed in Section 3. Section 4 studies some properties of equilibrium points of the proposed model. The main results of conditions enabling the proposed network to solve the CLM problem are given in Section 5. Section 6 presents a simulation example to further illustrate the theories. Finally, the conclusions are given in Section 7.

2 Competitive Layer Model

The CLM is a model of recurrently connected structure, as shown in [4]. It contains a set of L layers and there are N neurons in each layer. Neurons in each layer are connected with each other and the connection weights are independent of any layer. Each neuron has an external input which is also independent of any layer. Between the different layers, only those neurons that are arranged in the same row are connected. Neurons in the same layer are required to be cooperative, while neurons in the same row are required to be competitive. At the result of competition, there is only one active neuron in each row.

Throughout this paper, the symbols $\alpha, \beta, \gamma (1 \leq \alpha, \beta, \gamma \leq L)$, called layer indexes, will be used to index the layers, and the symbols $i, j (1 \leq i, j \leq N)$, called row indexes, will be used to index the rows. Thus, the index $i\alpha$ indicates that the neuron is located at the i th row of the α th layer. Let $x_{i\alpha}$ be the activity of the neuron located at $i\alpha$, w_{ij} be the connection weight between neuron $i\alpha$ and neuron $j\alpha$, note that each w_{ij} is independent of the layer index α and $w_{ii} > 0$. The external input for neuron $i\alpha$ is denoted by h_i (assumed $h_i > 0$) which is also independent of the layer index.

Denote the nonnegative orthant of R^{NL} by

$$R_+^{NL} = \{x \mid x \in R^{NL}, x \geq 0\},$$

where $x \geq 0$ implies that each element of the vector x is nonnegative. The CLM energy function $E(x)$ is defined in [4] by

$$E(x) = \frac{C}{2} \sum_{i=1}^N \left(\sum_{\beta=1}^L x_{i\beta} - h_i \right)^2 - \frac{1}{2} \sum_{\alpha=1}^L \sum_{i,j=1}^N w_{ij} x_{i\alpha} x_{j\alpha}$$

for $x \in R_+^{NL}$. Solving the CLM problem means to find out the minimum points of $E(x)$. By the rigorously mathematical analysis, [4] has shown that $x^* \in R_+^{NL}$ is a minimum points of $E(x)$, if and only if given any row index $i (1 \leq i \leq N)$, there exists one and only one layer index $\alpha (1 \leq \alpha \leq L)$ such that

$$\begin{cases} x_{i\alpha}^* > 0, \\ x_{i\gamma}^* = 0, \text{ for } \gamma \neq \alpha, \end{cases} \quad (1)$$

and

$$\begin{cases} C(h_i - x_{i\alpha}^*) + \sum_{j=1}^N w_{ij} x_{j\alpha}^* = 0, \\ C(h_i - x_{i\alpha}^*) + \sum_{j=1}^N w_{ij} x_{j\gamma}^* < 0, \text{ for } \gamma \neq \alpha. \end{cases} \quad (2)$$

Therefore, the study in [4] presented a precise instruction of implementing the CLM by designing some RNNs. That is, the trajectories of a designed RNN should converge to the minimum points of the CLM energy function $E(x)$, i.e., the conditions for the attractors in the designed RNN should match (1) and (2).

3 Discrete-Time Linear Threshold Recurrent Neural Networks

The model of discrete-time linear threshold recurrent neural networks that can be used to implement the CLM is described by

$$x_{i\alpha}(k+1) = \left[C \left(h_i - \sum_{\beta=1}^L x_{i\beta}(k) \right) + \sum_{j=1}^N w_{ij} x_{j\alpha}(k) + x_{i\alpha}(k) \right]^+, \quad k \geq 0 \quad (3)$$

for $1 \leq i \leq N$ and $1 \leq \alpha \leq L$. $[s]^+ = \max\{s, 0\}$ is a linear threshold function. $x(k) \in R^{NL}$ denotes the state of the network (3) at time k , it can be written as

$$x(k) = (x_1^T(k), \dots, x_L^T(k))^T \in R^{NL}$$

with

$$x_\alpha(k) = (x_{1\alpha}(k), \dots, x_{N\alpha}(k))^T \in R^N, \quad (1 \leq \alpha \leq L),$$

and each vector $x_\alpha(k) \in R^N$ is called a layer of the state $x(k)$.

In this paper, denote by I the identity matrix. Denote by $W = (w_{ij})_{N \times N}$, and

$$\bar{h} = \max_{1 \leq i \leq N} \{h_i\}, \quad \underline{h} = \min_{1 \leq i \leq N} \{h_i\}.$$

The trajectories of the network (3) are expected to converge to minimum points of the CLM energy function in R_+^{NL} space. Thus, it is a basic requirement that any trajectory of the network (3) starting from a point in R_+^{NL} remains in R_+^{NL} for ever. It is easy to see that the requirement is satisfied. Hence, in the next of this paper, it considers only the trajectories starting from the points in R_+^{NL} .

4 Equilibrium Points Analysis

This section studies the properties and distribution of equilibrium points of the network (3) in R_+^{NL} . The equilibrium points of the network (3) are some special vectors in R_+^{NL} space, any trajectory starting from an equilibrium point stays at the point all the time.

Definition 1. Suppose that $x^* \in R_+^{NL}$. If it holds that

$$x_{i\alpha}^* = \left[C \left(h_i - \sum_{\beta=1}^L x_{i\beta}^* \right) + \sum_{j=1}^N w_{ij} x_{j\alpha}^* + x_{i\alpha}^* \right]^+$$

for $1 \leq i \leq N$ and $1 \leq \alpha \leq L$, then x^* is called an equilibrium point of the network (3). Denote by \mathcal{E} the set of all the equilibrium points in R_+^{NL} .

Lemma 1. If $x^* \in \mathcal{E}$, then it holds that

$$\begin{cases} C \left(h_i - \sum_{\beta=1}^L x_{i\beta}^* \right) + \sum_{j=1}^N w_{ij} x_{j\alpha}^* = 0, & \text{if } x_{i\alpha}^* > 0 \\ C \left(h_i - \sum_{\beta=1}^L x_{i\beta}^* \right) + \sum_{j=1}^N w_{ij} x_{j\alpha}^* \leq 0, & \text{if } x_{i\alpha}^* = 0 \end{cases} \quad (4)$$

for $1 \leq i \leq N$ and $1 \leq \alpha \leq L$.

Proof. The proof is trivial.

The equilibrium points are distributed in the space R_+^{NL} . Next, we show that under some conditions, all the equilibrium points are located within a bounded set in the space R_+^{NL} . Denote

$$M = \max_{1 \leq i \leq N} \left\{ \sum_{j=1}^N |w_{ij}| \right\} \quad \text{and} \quad \Pi = \frac{C\bar{h}}{C - M}.$$

Lemma 2. Suppose that $C > M$. Define a set by

$$D = \{ x \in R_+^{NL} \mid x_{j\beta} \leq \Pi, (1 \leq j \leq N; 1 \leq \beta \leq L) \}.$$

Then, it holds that $\mathcal{E} \subseteq D$.

Proof. Given any $x^* \in \mathcal{E}$, suppose that

$$x_{i\alpha}^* = \max \{ x_{j\beta}^* \mid 1 \leq j \leq N; 1 \leq \beta \leq L \}.$$

If $x_{i\alpha}^* = 0$, then clearly, $x^* \in D$. If $x_{i\alpha}^* > 0$, by (4), it must hold that

$$\begin{aligned} 0 &= C \left(h_i - \sum_{\beta=1}^L x_{i\beta}^* \right) - \sum_{j=1}^N w_{ij} x_{j\alpha}^* \\ &\leq C(\bar{h} - x_{i\alpha}^*) + x_{i\alpha}^* \cdot M \\ &= C\bar{h} - x_{i\alpha}^*(C - M). \end{aligned}$$

It gives that

$$x_{j\beta}^* \leq x_{i\alpha}^* \leq \Pi$$

for $1 \leq j \leq N$ and $1 \leq \beta \leq L$, i.e., $x^* \in D$. The proof is complete.

Theorem 1. Suppose that

$$C > \left(\frac{\bar{h}}{h} + 1 \right) \cdot M.$$

If $x^* \in \mathcal{E}$, then given any row index i ($1 \leq i \leq N$), there must exist at least one layer index α ($1 \leq \alpha \leq L$) such that $x_{i\alpha}^* > 0$.

Proof. Given any row index $1 \leq i \leq N$, suppose that $x_{i\alpha}^* = 0$ for all $1 \leq \alpha \leq L$. It follows from (4) that

$$\begin{aligned} 0 &\geq C \left(h_i - \sum_{\beta=1}^L x_{i\beta}^* \right) + \sum_{j=1}^N w_{ij} x_{j\alpha}^* \\ &= Ch_i + \sum_{j=1}^N w_{ij} x_{j\alpha}^* \end{aligned}$$

for $1 \leq \alpha \leq L$. However, by the condition and Lemma 2, it gives that

$$Ch_i + \sum_{j=1}^N w_{ij} x_{j\alpha}^* \geq Ch - \Pi \cdot M > 0$$

for all $1 \leq \alpha \leq L$. This contradiction shows that there must exist a layer index α such that $x_{i\alpha}^* > 0$. The proof is complete.

5 Stable Attractors

Since it is expected to use the network (3) to find out the minimum points of the CLM energy function $E(x)$ in R_+^{NL} , the trajectories of the network (3) should be able to converge to the minimum points of $E(x)$. Generally, a point in R_+^{NL} is called a stable attractor if it attracts all trajectories starting from the points around it. The following discussion will show that the stable attractors of the network (3) correspond to the minimum points of the CLM energy function.

Definition 2. Let x^* be an equilibrium point. The equilibrium point x^* is called stable, if given any constant $\epsilon > 0$, there exists a constant $\delta > 0$ such that

$$|x_{i\alpha}(0) - x_{i\alpha}^*| \leq \delta, \quad (1 \leq i \leq N; 1 \leq \alpha \leq L)$$

imply that

$$|x_{i\alpha}(k) - x_{i\alpha}^*| \leq \epsilon, \quad (1 \leq i \leq N; 1 \leq \alpha \leq L)$$

for all $k \geq 0$. The equilibrium point x^* is called unstable, if it is not stable.

Definition 3. Let x^* be a stable equilibrium point, it is called a stable attractor, if there exists a small neighborhood B of x^* such that $x(0) \in B$ implies that

$$\lim_{k \rightarrow +\infty} x_{i\alpha}(k) = x_{i\alpha}^*$$

for $1 \leq i \leq N$ and $1 \leq \alpha \leq L$.

A stable attractor is an equilibrium point with the properties: it should be stable in the sense of Definition 2, and it must attract the trajectories around it.

Theorem 2. Suppose that

$$C > \left(\frac{\bar{h}}{\underline{h}} + 1 \right) \cdot M \quad (5)$$

and

$$C < 2 + \min_{1 \leq i \leq N} \left\{ w_{ii} - \sum_{j=1, j \neq i}^N |w_{ij}| \right\}. \quad (6)$$

The equilibrium point x^* of the network (3) is a stable attractor, if it satisfies that (1) and (2).

Proof. Denote two index sets by

$$\begin{cases} P = \{i\alpha | x_{i\alpha}^* > 0; 1 \leq i \leq N; 1 \leq \alpha \leq L\} \\ Z = \{i\gamma | x_{i\gamma}^* = 0; 1 \leq i \leq N; 1 \leq \gamma \leq L\}. \end{cases}$$

By (2), given any index $i\gamma \in Z$, denote a constant $\Delta_{i\gamma}$ by

$$\Delta_{i\gamma} = -C(h_i - x_{i\alpha}^*) - \sum_{j=1}^N w_{ij} x_{j\gamma}^* > 0.$$

It follows that

$$C(h_i - x_{i\alpha}^*) + \sum_{j=1}^N w_{ij} x_{j\gamma}^* + \Delta_{i\gamma} = 0.$$

Then the network (3) can be rewritten as

$$\begin{cases} x_{i\alpha}(k+1) = \left[-C \sum_{\beta=1}^L (x_{i\beta}(k) - x_{i\beta}^*) \right. \\ \quad \left. + \sum_{j=1}^N w_{ij} (x_{j\alpha}(k) - x_{j\alpha}^*) + (x_{i\alpha}(k) - x_{i\alpha}^*) + x_{i\alpha}^* \right]^+, i\alpha \in P \\ x_{i\gamma}(k+1) = \left[-C \sum_{\beta=1}^L (x_{i\beta}(k) - x_{i\beta}^*) \right. \\ \quad \left. + \sum_{j=1}^N w_{ij} (x_{j\gamma}(k) - x_{j\gamma}^*) + x_{i\gamma}(k) - \Delta_{i\gamma} \right]^+, i\gamma \in Z \end{cases}$$

for all $k \geq 0$.

Given any constant ϵ such that

$$0 < \epsilon \leq \frac{1}{3} \cdot \min_{i\alpha \in P, i\gamma \in Z} \{x_{i\alpha}^*, \Delta_{i\gamma}\} \cdot \min \left\{ \frac{1}{CL}, \frac{1}{M+1} \right\},$$

define a constant δ by

$$\delta = \frac{\epsilon}{3} \cdot \min \left\{ \frac{1}{CL}, \frac{1}{M+1} \right\}.$$

Then define a neighborhood B_δ of x^* by

$$B_\delta = \left\{ x \in R_+^{NL} \mid |x_{i\alpha} - x_{i\alpha}^*| \leq \delta (i\alpha \in P); x_{i\gamma} \leq \delta (i\gamma \in Z) \right\},$$

and another neighborhood B_ϵ of x^* by

$$B_\epsilon = \left\{ x \in R_+^{NL} \mid |x_{i\alpha} - x_{i\alpha}^*| \leq \epsilon (i\alpha \in P); x_{i\gamma} = 0 (i\gamma \in Z) \right\}.$$

We will prove by mathematical induction that for any $x(0) \in B_\delta$, it holds that $x(k) \in B_\epsilon$ for all $k \geq 1$. By $x(0) \in B_\delta$, we have

$$x_{i\gamma}(1) = \left[-C \sum_{\beta=1}^L (x_{i\beta}(0) - x_{i\beta}^*) + \sum_{j=1}^N w_{ij} (x_{j\gamma}(0) - x_{j\gamma}^*) + x_{i\gamma}(0) - \Delta_{i\gamma} \right]^+ = 0$$

for $i\gamma \in Z$, and

$$\begin{aligned} x_{i\alpha}(1) &= \left[-C \sum_{\beta=1}^L (x_{i\beta}(0) - x_{i\beta}^*) + \sum_{j=1}^N w_{ij} (x_{j\alpha}(0) - x_{j\alpha}^*) + (x_{i\alpha}(0) - x_{i\alpha}^*) + x_{i\alpha}^* \right]^+ \\ &= -C \sum_{\beta=1}^L (x_{i\beta}(0) - x_{i\beta}^*) + \sum_{j=1}^N w_{ij} (x_{j\alpha}(0) - x_{j\alpha}^*) + (x_{i\alpha}(0) - x_{i\alpha}^*) + x_{i\alpha}^* \end{aligned}$$

for $i\alpha \in P$. Then,

$$\begin{aligned} |x_{i\alpha}(1) - x_{i\alpha}^*| &\leq C \sum_{\beta=1}^L |x_{i\beta}(0) - x_{i\beta}^*| + \sum_{j=1}^N |w_{ij}| \cdot |x_{j\alpha}(0) - x_{j\alpha}^*| + |x_{i\alpha}(0) - x_{i\alpha}^*| \\ &\leq \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon. \end{aligned}$$

This shows that $x(1) \in B_\epsilon$. Next, suppose $x(k) \in B_\epsilon$, we will prove that $x(k+1) \in B_\epsilon$ for all $k \geq 1$. In fact, it follows from $x(k) \in B_\epsilon$ that

$$\begin{aligned} x_{i\gamma}(k+1) &= \left[-C (x_{i\alpha}(k) - x_{i\alpha}^*) + \sum_{j=1, j\gamma \in P}^N w_{ij} (x_{j\gamma}(k) - x_{j\gamma}^*) - \Delta_{i\gamma} \right]^+ \\ &= 0 \end{aligned} \tag{7}$$

and

$$\begin{aligned} x_{i\alpha}(k+1) &= \\ &= \left[-C (x_{i\alpha}(k) - x_{i\alpha}^*) + \sum_{j=1, j\alpha \in P}^N w_{ij} (x_{j\alpha}(k) - x_{j\alpha}^*) + (x_{i\alpha}(k) - x_{i\alpha}^*) + x_{i\alpha}^* \right]^+ \\ &= -C (x_{i\alpha}(k) - x_{i\alpha}^*) + \sum_{j=1, j\alpha \in P}^N w_{ij} (x_{j\alpha}(k) - x_{j\alpha}^*) + (x_{i\alpha}(k) - x_{i\alpha}^*) + x_{i\alpha}^* \end{aligned}$$

for all $k \geq 1$. Then,

$$\begin{aligned} x_{i\alpha}(k+1) - x_{i\alpha}^* &= -C(x_{i\alpha}(k) - x_{i\alpha}^*) + (x_{i\alpha}(k) - x_{i\alpha}^*) \\ &\quad + \sum_{j=1, j\alpha \in P}^N w_{ij} (x_{j\alpha}(k) - x_{j\alpha}^*) \end{aligned} \quad (8)$$

for $k \geq 1$. Since given any row index there exists one and only one layer index such that the indexed neuron is active, the set P has N elements in total. Denote by $x_P \in R_+^N$ and $x_P^* \in R_+^N$ the vectors which are composed of the elements $x_{i\alpha}$ and $x_{i\alpha}^*$, respectively. Then it follows from Eq. (8) that

$$x_P(k+1) - x_P^* = \Phi \cdot (x_P(k) - x_P^*),$$

where the matrices $\Phi = (1 - C)I + W_P$ with $W_P = (v_{ij})_{N \times N}$ and

$$\begin{cases} v_{ij} = w_{ij}, & \text{if } i\alpha, j\alpha \in P, \\ v_{ij} = 0, & \text{if } i\alpha \notin P \text{ or } j\alpha \notin P. \end{cases}$$

Then it holds that

$$\|x_P(k+1) - x_P^*\| \leq \max \{|\lambda|\} \cdot \|x_P(k) - x_P^*\|, \quad (9)$$

where λ is the eigenvalue of Φ . By applying Gershgorin's Theorem [8], [5] and [6] imply that $\max \{|\lambda|\} < 1$. Then (7) and (9) show that $x(k+1) \in B_\epsilon$ for all $k \geq 1$.

Thus, any trajectory $x(k)$ starting from a point $x(0) \in B_\delta$ will stay in B_ϵ forever. Moreover, because of $\max \{|\lambda|\} < 1$, it is clear that

$$\begin{cases} x_{i\alpha}(k) \rightarrow x_{i\alpha}^* \\ x_{i\gamma}(k) \rightarrow 0 \end{cases}$$

as $k \rightarrow +\infty$. It shows that x^* is a stable attractor. The proof is complete.

By the necessary and sufficient conditions (1) and (2) for the minimum points of CLM energy function $E(x)$, Theorem 2 shows that the stable attractors of the network (3) correspond to the minimum points of $E(x)$ on the nose. Therefore, the proposed model of discrete-time LT RNNs can be used to solve the CLM problem.

6 Illustrative Example

To further understand the attractors of the network (3), consider the case that $L = 2$, $N = 2$, $w_{ij} = w > 0$ ($i, j = 1, 2$) and $h_i = h > 0$ ($i = 1, 2$). The network is

$$\begin{cases} x_{11}(k+1) = [C(h - x_{11}(k) - x_{12}(k)) + w(x_{11}(k) + x_{21}(k)) + x_{11}(k)]^+ \\ x_{21}(k+1) = [C(h - x_{21}(k) - x_{22}(k)) + w(x_{11}(k) + x_{21}(k)) + x_{21}(k)]^+ \\ x_{12}(k+1) = [C(h - x_{11}(k) - x_{12}(k)) + w(x_{12}(k) + x_{22}(k)) + x_{12}(k)]^+ \\ x_{22}(k+1) = [C(h - x_{21}(k) - x_{22}(k)) + w(x_{12}(k) + x_{22}(k)) + x_{22}(k)]^+ \end{cases} \quad (10)$$

for $k \geq 0$.

Suppose that $C = 0.9$, $w = 0.2$ and $h = 1$, which satisfy (5) and (6). Solving the equilibrium point of the network (10), it gives that

$$\mathcal{E} = \{(1.8, 1.8, 0, 0)^T, (0, 0, 1.8, 1.8)^T\}.$$

By Theorem 2, it can be check that the two equilibrium points above are both stable attractors of the network (10). Moreover, given any row index, there is one and only one active neuron on the row. Fig. II shows the attractivity of the two stable attractors, respectively.

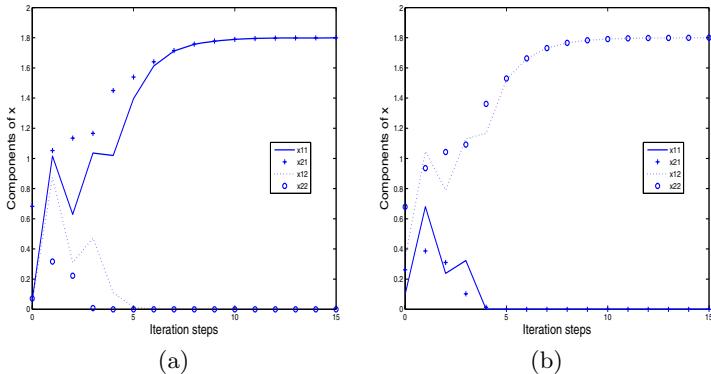


Fig. 1. The attractivity of attractors in the network (10): (a) The trajectory starting from a randomly generated point $x(0) = (0.0596, 0.6820, 0.0424, 0.0714)^T$ converges to $(1.8, 1.8, 0, 0)^T$; (b) The trajectory starting from another randomly generated point $x(0) = (0.0987, 0.2619, 0.3354, 0.6797)^T$ converges to $(0, 0, 1.8, 1.8)^T$

7 Conclusions

The paper induces the discrete-time LT RNNs enabling to implement CLM. It gives some properties of the equilibrium points of this model. Furthermore, the conditions for stable attractors of the networks are established based on the rigorously mathematical analysis, which just match those of the minimum points of CLM energy function established before. Therefore, the proposed model can be used to solve the CLM problem. The study of this paper enables a perspective to implementing CLM by the discrete-time LT RNNs for the potential applications.

Acknowledgement

This work was supported by Chinese 863 High-Tech Program under Grant 2007AA01Z321.

References

1. Ritter, H.: A Spatial Approach for Feature Linking. In: Proc. Int. Neural Networks Conf., Paris, vol. 2, pp. 898–901 (1990)
2. Von der Malsburg, C.: Binding in Models of Perception and Brain Function. Current Opinion in Neurobiology 5, 520–526 (1995)
3. Wersing, H., Steil, J.J., Ritter, H.: A Competitive Layer Model for Feature Binding and Sensory Segmentation. Neural Computation 13, 357–387 (2001)
4. Yi, Z.: Foundations of Implementing the Competitive Layer Model by Lotka-Volterra Recurrent Neural Networks. Conditionally Accepted by IEEE Trans. Neural Networks (2009)
5. Zhang, L., Yi, Z., Yu, J.: Multiperiodicity and Attractivity of Delayed Recurrent Neural Networks with Unsaturating Piecewise Linear Transfer Functions. IEEE Trans. Neural Networks 19, 158–167 (2008)
6. Yu, J., Yi, Z., Zhang, L.: Representations of Continuous Attractors of Recurrent Neural Networks. IEEE Trans. Neural Networks 20, 368–372 (2009)
7. Yi, Z., Tan, K.K.: Multistability Analysis of Discrete Recurrent Neural Networks with Unsaturating Piecewise Linear Transfer Functions. IEEE Trans. Neural Networks 15, 329–336 (2004)
8. Goldberg, J.L.: Matrix Theory with Applications. McGraw-Hill, New York (1992)

Scalable Neural Networks for Board Games

Tom Schaul and Jürgen Schmidhuber

IDSIA, Galleria 2, Manno-Lugano, Switzerland

{tom,juergen}@idsia.ch

Abstract. Learning to solve small instances of a problem should help in solving large instances. Unfortunately, most neural network architectures do not exhibit this form of scalability. Our Multi-Dimensional Recurrent LSTM Networks, however, show a high degree of scalability, as we empirically show in the domain of flexible-size board games. This allows them to be trained from scratch up to the level of human beginners, without using domain knowledge.

1 Introduction

In a wide range of domains it is possible to learn from a simple version of a problem and then use this knowledge on a larger one. This particular form of incremental learning is commonly employed by humans, and for machine learning it is especially useful when training on the large version is much more expensive.

Board games are a particularly suitable domain for investigating this form of *scalability*, because for many of them either the board size can be varied, or the rules can be trivially adjusted to make it variable. In addition, despite being described by a small set of formal rules, they often involve highly complex strategies. One of the most interesting board games is the ancient game of *Go* (among other reasons, because computer programs are still much weaker than human players), which can be solved for small boards [1] but is very challenging for larger ones [23]. Its extremely large search space defies traditional search-based methods. Human experts rely heavily on patterns, and thus it is not surprising that a substantial amount of research effort has been devoted to applying neural networks – which are good at pattern recognition – to Go [24]. Unfortunately most of these methods do not scale well w.r.t. board size, i.e. networks trained successfully on small boards (where training is efficient) do not play well when the board is enlarged [53]. The present paper builds on the promising preliminary results [6,7] of a scalable approach based on Multi-dimensional Recurrent Neural Networks (MDRNNs; [8,9]) and enhances the ability of that architecture to capture long-distance dependencies.

We conduct experiments on three different Go-inspired games, which is possible without modifying our network architecture as it is free of *domain knowledge*. We train it against opponents of varying difficulty and measure how the playing performance scales to larger board sizes. Furthermore, we put our architecture into context by comparing it to a number of competing ones.

2 Scalable Neural Architectures

We consider a neural network architecture to be scalable if it is not tied to a fixed input dimension. This section provides an overview of such architectures that have been proposed for solving board games, it then describes MDRNNs in general and finally gives the details of the specific instantiation we propose.

2.1 Background

One approach to designing scalable network architectures is to scan across the board, processing the inputs from a limited *receptive field*, independently of their positions. The outputs of that stage are then fed into another architecture that combines them (e.g. [10]). An extension of this idea is the *convolutional network* [11], which repeats this step on multiple levels, thereby capturing higher-level features instead of just local patterns. These architectures introduce a trade-off: a small receptive field severely limits the kind of patterns that can be recognized, whereas a large one makes learning very difficult (because of the exploding number of parameters).

‘Roving-eye’-based architectures [5] contain one component with a fixed receptive field that can be aimed at any part of the board. This is then combined with an active component that decides where to rove over the board, and when to choose an output action.

Other architectures have been proposed [12, 13] which make use of weight-sharing to capture domain-specific symmetries, but these are limited to a particular game, and also restricted w.r.t. what kind of strategies they can learn.

For the related but different problem of scaling the problem *resolution*, a number of approaches for generative encodings of neural networks have been found to be successful (e.g. Compositional Pattern Producing Networks [14]).

2.2 MDRNNs

Multi-dimensional Recurrent Neural Networks [8, 9], are an extension of bi-directional RNN proposed by Schuster [15], and a special case of the DAG-RNNs proposed by Baldi [16]. Their unbounded receptive fields (explained below) make them scalable by design. Successful applications include vision [8], handwriting recognition [17], and supervised learning of expert Go moves [4].

Unlike standard recurrent neural networks (RNNs) which are only effective for handling sequences with a single (time-)dimension, MDRNNs are applicable to multi-dimensional sequences [9]. In the case of Go, the single time dimension is replaced by the two space dimensions of the game board.

Consider a hidden layer h_{\nearrow} that *swipes* diagonally over the board from bottom-left to top-right. At each board position (i, j) its activation $h_{\nearrow(i,j)}$ is a function of the current input $in_{i,j}$ and its own earlier activations $h_{\nearrow(i-1,j)}$ and $h_{\nearrow(i,j-1)}$:

$$h_{\nearrow(i,j)} = f(w_i * in_{i,j} + w_h * h_{\nearrow(i-1,j)} + w_h * h_{\nearrow(i,j-1)}) \quad (1)$$

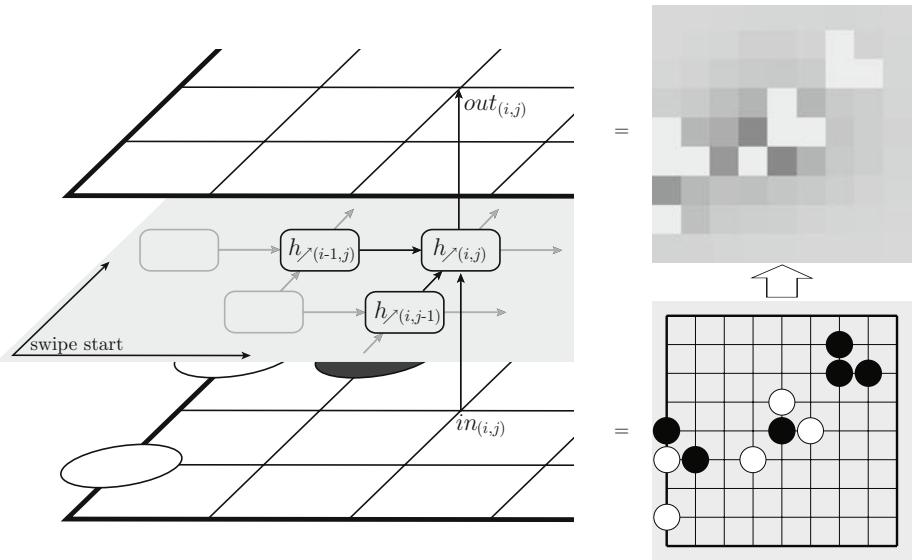


Fig. 1. MDRNN for Go The structure diagram on the left shows the connections of a hidden layer in one of the swiping directions: it receives its two earlier activations, as well as the local board input. The network as a whole takes the game board as input (bottom right) and outputs move preferences (top right). The darker the square, the higher the preference for the corresponding move (illegal ones are ignored).

where w_* are the connection weights. On the boundaries we use a fixed default value $h_{\nearrow(0,i)} = h_{\nearrow(i,0)} = w_b$, for $0 < i \leq n$. See also Figure 1 for an illustration.

Because of the recurrency, the layer has indirect access to board information from the whole rectangle between $(0, 0)$ and (i, j) . In order to have access to the whole board, we use 4 swiping layers, one for each of the diagonal swiping directions in $D = \{\nwarrow, \nearrow, \swarrow, \nwarrow\}$. The output layer then, for every position, combines the outputs of these 4 layers into a single value $out_{i,j}$ (which is indirectly derived from the information of the entire input). More formally:

$$out_{i,j} = g \left(\sum_{\diamond \in D} w_o * h_{\diamond(i,j)} \right) \quad (2)$$

where the function g is typically the sigmoid function.

2.3 Proposed Architecture

We instantiate MDRNNs such that they are appropriately generating a playing policy, given a symmetric game board. At each position, the network takes two *inputs* which indicate the presence of a stone at this position. The first one is 1 if a stone of the network's own color is present and 0 otherwise, the second input encodes the presence of an opponent's stone in the same way. A black/white

symmetric encoding, as used in other approaches (e.g. [12]) is not applicable here, because the output is not symmetrical: the best move for both players might be the same.

The *output* value at each position expresses the network’s *preference* for playing there (see also Figure 1). Assuming that a deterministic playing behavior is desired, moves are selected greedily, randomly breaking ties. This is the case in our experiments because the opponents act stochastically. In practice, we ignore the network’s preferences for illegal moves. For stochastic play one can interpret the preferences probabilistically, e.g. by drawing a position from their Gibbs distribution.

MDRNNs are invariant w.r.t. stationary shifts of the input. In order to also enforce rotation and reflection symmetry, we use the same connection weights for all swiping directions and the same w_b on all boundaries.

Typically a swiping layer u is composed of k sigmoidal neurons (e.g. $f = \tanh$). Although in theory such an MDRNN can learn to make use of the whole board context, it is very difficult to achieve in practice, because the information is propagated recurrently through non-linear units and thus tends to decay quickly with distance [18]. One solution to this problem is to use *Long short-term memory* cells (LSTM), which are based on protecting the recurrent state with *gating* sub-units [18]. As in [89] (and in contrast to [6]), we therefore use a swiping layer composed of k LSTM cells and call it *MDLSTM*.

In our implementation we unfold the MDRNN along both spacial dimensions, leading to a large but simple feed-forward network. On a normal desktop computer (Intel Xeon 2.8 GHz), a network needs about 3ms to choose a move on a 7x7 board, and 25ms on a 19x19 board. The total number of weights is $4k + k^2$ for sigmoidal swiping layers and $5k^2 + 16k$ for LSTM layers. All the code used for this paper is available as part of the PyBrain library at www.pybrain.org.

3 Methodology

We conduct our experiments on a number of different flexible-size board games, all of which are played on the Go board:

Atari-Go: Also known as Ponnuki-Go or ‘Capture Game’, is a simplified version of Go that is widely used for teaching the game of Go to new players. The rules are the same as for Go, except that passing is not allowed, and the first player to capture a predetermined number (here: one) of his opponent’s stones wins.

Go-Moku: Is also known as ‘Five-in-a-row’. Players alternate putting stones onto any of the intersections on the board. The first player to have 5 connected stones in a row, column or diagonal, wins.

Pente: Has similar rules to Go-Moku, except that it is now possible to capture stones, in pairs, by putting stones at both ends of a pair of the opponent. The game is won by the first player who either has 5 connected stones, or has captured 5 pairs.

Each game has a number of predefined opponents associated with it: *a*) a *random* player, which randomly chooses any of the legal moves, *b*) a *naive* player, which does a one-ply search. If possible, it always picks a move that makes it win the game immediately, and never picks a move that would make it lose the game immediately. In all other cases (the large majority), it randomly picks a legal move, *c*) a publicly available *heuristic* player (only for Atari-Go), based on a set of hand-coded heuristic tactics (exploited greedily [7,19]). Its difficulty can be adjusted by imposing that a proportion ϵ of its moves are chosen randomly. According to the author, the level of play (with $\epsilon = 0$) is ‘challenging for beginners’.

As *fitness* we use the average outcome of 100 games against a fixed opponent, counting a win as 1, a draw as 0 and a loss as -1. Each player plays 50 times as black and 50 times as white.

In addition to MDRNNs with sigmoidal neurons or LSTM cells (as described in section 2.3), we use – as a performance baseline – standard multi-layer perceptrons (MLP), containing a single fully connected hidden layer of size k , with tanh units. We compare the performance of our architecture to simple convolutional networks (CONV), with one layer of k feature maps (of identical receptive field size $\rho \times \rho$), no subsampling, and a sigmoid output layer that combines the features. Here, the input board is padded with additional positions around the borders. They have $k(\rho^2 + 1) + 1$ parameters.

One the one hand, we analyze the performance of networks produced by the simplest possible algorithm, namely random weight guessing (using the normal distribution $N(0, 1)$). On the other hand we train the networks using the well-established Covariance Matrix Adaptation Evolution Strategy (CMA-ES [20]) to optimize all the weights.

Our two quantitative measures of scalability are: *a*) the linear correlation (Pearson coefficient) between the fitness on different board sizes *b*) the proportion p of networks for which the fitness is higher on the larger board than on the smaller one.

4 Results

As training networks is expensive, we start by empirically investigating the performance of the different networks (and their parameters) with random weights. Moreover, we determine under what circumstances the performance scales to larger boards. We then train the networks on small boards and analyze whether their performance improvement is transferred to larger boards.

4.1 Random Networks

We measure the performance of different kinds of networks with randomly guessed weights, on different games and against various opponents. Figure 2(a) shows the percentiles of fitnesses of random MDRNNs, giving an indication of the difficulty of the different opponents on each game. Training is easier if initial weights with

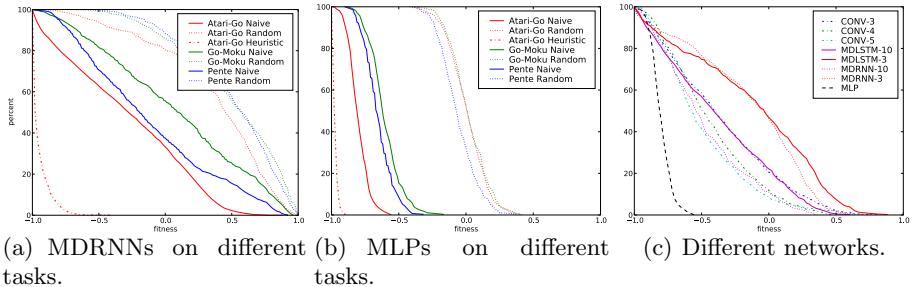


Fig. 2. Fitnesses of random networks evaluated on 7x7 (400 networks per scenario). The percentiles show what proportion of random networks reach at least a given fitness (e.g. at least 25% or random MDRNNs win at least $\frac{3}{4}$ of Go-Moku games against the naive opponent, i.e. have a fitness of 0.5). Figure 2(c) shows the differences between a number of different networks (on Atari-Go, vs. Naive).

reasonably good fitness (> 0) can be found relatively easily. This is indeed the case for the naive and the random opponent but not for the heuristic one. For MLPs, however, reasonable initial weights are very rare (Figure 2(b)).

Figure 2(c) more explicitly shows the differences between the network architectures, comparing MDRNNs, MDLSTMs (for varying k), CONVs (for varying ρ) and MLPs. Despite only corresponding to random networks, the results indicate that small values of k are appropriate for MDRNNs (we will fix $k = 3$ hereafter), and do not bode well for MLPs.

We determine the scalability of random networks by evaluating the fitness on multiple board sizes and then computing their correlation (see Table I). As the linear correlation by itself can be a misleading measure, we provide a visual intuition about the high correlation in Figure 4(a). The results indicate that one can train networks on boards as small as 7x7 and use them to play on 19x19, for all three games.

4.2 Trained Networks

Figure 3(a) shows the learning curves for different networks trained on Atari-Go against the naive player on 7x7, using CMA-ES to optimize the weights. MLPs are in this comparison as a baseline, but clearly fail to learn how to play. The other architectures learn to beat the naive opponent, with MDLSTMs clearly outperforming the others. Convolutional networks are learning slightly slower, but still faster than MDRNNs. The same conclusions also hold for the results on Go-Moku and Pente (results not shown).

Learning to play against the significantly stronger heuristic opponent is a bigger challenge. Figures 3(b) and 3(c) show the learning curves against the heuristic player with settings $\epsilon = 0.2$ and $\epsilon = 0.05$ respectively (averaged over 5 runs). Here, MDLSTMs clearly outperform the convolutional networks, for which only the best results are shown (produced with $\rho = 5$). We suspect that

Table 1. Correlations between the fitnesses of random MDLSTMs on different board sizes (based on 100 networks per scenario, evaluated against the naive opponent). They are high in all cases except Go-Moku between 5x5 and larger boards (which is due to the fact that it is disproportionately easier for a game to end in a draw on a 5x5 board).

Sizes	Atari-Go	Go-Moku	Pente
5x5 vs. 7x7	0.86	0.20	0.47
5x5 vs. 9x9	0.72	0.09	0.31
5x5 vs. 11x11	0.67	0.37	0.49
5x5 vs. 19x19	0.37	0.38	0.46
7x7 vs. 9x9	0.88	0.83	0.83
7x7 vs. 11x11	0.82	0.85	0.87
7x7 vs. 19x19	0.62	0.59	0.64
9x9 vs. 11x11	0.92	0.92	0.90
9x9 vs. 19x19	0.71	0.76	0.64

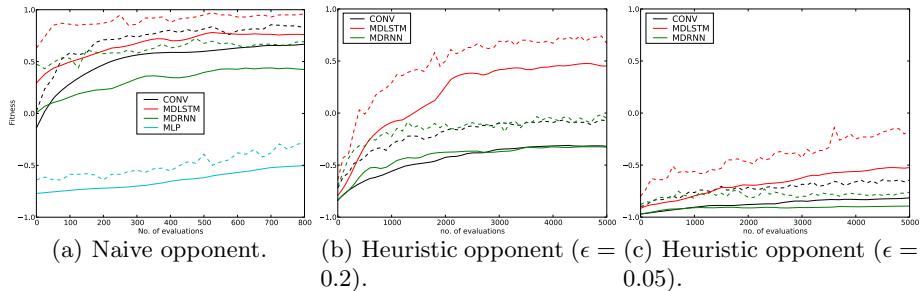


Fig. 3. Learning curves for training against different opponents on Atari-Go (board size 7x7, averaged over 10 independent runs). The solid line corresponds to the average fitness per generation, the broken one corresponds to the best.

this is due to the limited receptive field: at a certain level of play it simply becomes necessary to use non-local information. MDLSTMs can *learn* how much context is necessary, and automatically increase their effective receptive field during training.

The scalability results for trained networks are summarized in Table 2. Generally, there is a low correlation for the convolutional networks, but a relatively high one for MDLSTMs. Figure 4(b) illustrates this difference for networks trained against the naive opponent in Atari-Go. Note the large number of convolutional networks on the bottom right, for which good performance on 7x7 corresponds to poor performance on 11x11.

Comparing the correlations and the proportions p to the values for random MDLSTMs, we find the correlations to be lower but p to be higher (especially when scaling to 19x19). This means that the fitness on a small board is not perfectly predictive of the fitness on the large board, *but* it is almost always significantly higher on the large board.

Table 2. Scalability of networks trained on 7x7 against the naive opponent (based on 100 networks per scenario). The correlations are higher for MDLSTMs than for convolutional networks. Also, note the really high proportion p of MDLSTMs that are stronger on 19x19 than on 7x7, for all games.

Game	Sizes	CONV		MDLSTM	
		Correlation	p	Correlation	p
Atari-Go	7x7 vs. 9x9	0.13	0.20	0.38	0.48
Atari-Go	7x7 vs. 11x11	0.17	0.18	0.27	0.45
Atari-Go	7x7 vs. 19x19	0.17	0.21	0.38	0.76
Go-Moku	7x7 vs. 9x9	0.06	0.42	0.47	0.67
Go-Moku	7x7 vs. 11x11	0.15	0.61	0.38	0.87
Go-Moku	7x7 vs. 19x19	0.04	0.68	0.66	0.84
Pente	7x7 vs. 9x9	0.05	0.45	0.08	0.61
Pente	7x7 vs. 11x11	0.24	0.58	0.39	0.79
Pente	7x7 vs. 19x19	0.23	0.58	-0.05	0.95

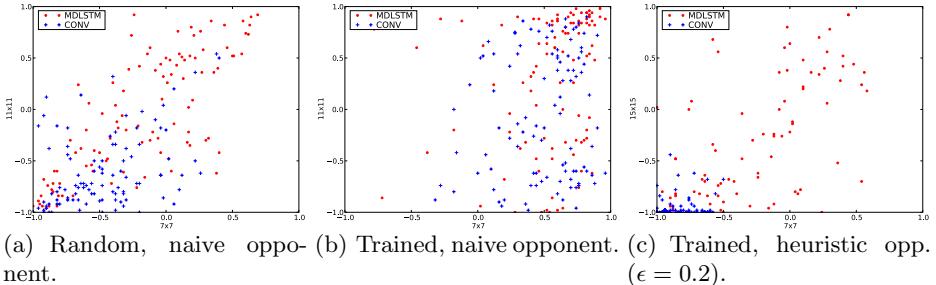


Fig. 4. Illustrations of fitness scalability on Atari-Go. The points correspond to MDLSTMs, the crosses to convolutional networks.

Of particular interest is the scalability of the networks trained against the strongest opponent – as figure 4(c) illustrates (for 7x7 versus 15x15), MDLSTMs achieve a high correlation (0.64), unlike convolutional networks (0.08).

5 Discussion

MDRNNs are scalable because they are approximately translation-invariant, in addition to capturing the board symmetries. They handle the same situation on a bigger board (with empty positions around it) in the exact same way as on a smaller board. This allows them to use a comparatively low number of weights (MDRNNs and MDLSTMs with $k = 3$ have 21 and 93 weights respectively, independently of board size), thereby reducing the dimensionality of the search space and making training efficient. Incorporating LSTM cells in the swiping layer further enabled the architecture to better handle long-distance context, which is necessary for learning complex strategies, as our experiments versus the

heuristic opponent show. Finally, the results show that MDLSTMs transfer the strategies learned on small boards to large ones, leading to a level of play on 15x15 that is on par with human beginners.

Directly training against a stronger opponent (e.g. $\epsilon = 0.0$) is like finding a needle in a haystack, as almost all initial networks will lose all games. In future work we will attempt to address this issue by training against incrementally harder opponents [21]. We expect to eventually reach a limit on the complexity of strategies that MDLSTMs can represent. In that case we propose increasing their representative power by stacking two (or more) MDLSTMs on top of each other, the lower one producing a map of high-level features that the top one scans over. MDLSTMs performed equally well on a number of different games, which we attribute to them being free from domain knowledge. However, in case performance is preferred over generality, our approach can easily be extended to make use of such knowledge, e.g. by feeding the network a number of domain-specific features [4] instead of the raw board. Due to the generality of our approach and the similarity of our three games to Go, we expect our positive results to carry over the game of Go itself, which we intend to investigate as our next step.

6 Conclusion

We have developed and investigated the properties of MDLSTMs, a scalable neural network architecture based on MDRNNs and LSTM cells. By validating our results on three different games, we showed that it is suitable for the domain of board games. Further, we found that training lead to an impressive level of play, given that they are devoid of domain knowledge and learn the games from scratch. Finally, the networks trained on small boards scale very well to large ones.

Acknowledgments

This research was funded by the SNF grant 200021-113364/1. We would like to thank Mandy Grüttner for building the framework to interact with the heuristic player, Justin Bayer for speeding up PyBrain to such a level that our extensive experiments became feasible, as well as Faustino Gomez for the constructive advice.

References

1. van der Werf, E., van den Herik, H.J., Uiterwijk, J.: Solving Go on small boards. International Computer Games Association Journal 26 (2003)
2. Richards, N., Moriarty, D.E., Miikkulainen, R.: Evolving neural networks to play Go. Applied Intelligence 8, 85–96 (1997)
3. Runarsson, T.P., Lucas, S.M.: Co-evolution versus Self-play Temporal Difference Learning for Acquiring Position Evaluation in Small-board Go. IEEE Transactions on Evolutionary Computation, 628–640 (2005)

4. Wu, L., Baldi, P.: A Scalable Machine Learning Approach to Go. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) *Advances in Neural Information Processing Systems 19*, pp. 1521–1528. MIT Press, Cambridge (2007)
5. Stanley, K.O., Miikkulainen, R.: Evolving a Roving Eye for Go. In: Deb, K., et al. (eds.) *GECCO 2004*. LNCS, vol. 3103, pp. 1226–1238. Springer, Heidelberg (2004)
6. Schaul, T., Schmidhuber, J.: A Scalable Neural Network Architecture for Board Games. In: *Proceedings of the IEEE Symposium on Computational Intelligence in Games*. IEEE Press, Los Alamitos (2008)
7. Grüttner, M.: Evolving Multidimensional Recurrent Neural Networks for the Capture Game in Go. Bachelor Thesis, Technische Universität München (2008)
8. Graves, A., Fernández, S., Schmidhuber, J.: Multidimensional Recurrent Neural Networks. In: *Proceedings of the 2007 International Conference on Artificial Neural Networks* (September 2007)
9. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks, Ph.D. in Informatics, Fakultat für Informatik – Technische Universität München (2008)
10. Silver, D., Sutton, R.S., Müller, M.: Reinforcement Learning of Local Shape in the Game of Go. In: *IJCAI*, pp. 1053–1058 (2007)
11. Lecun, Y., Bengio, Y.: Convolutional Networks for Images, Speech and Time Series, pp. 255–258. The MIT Press, Cambridge (1995)
12. Schraudolph, N.N., Dayan, P., Sejnowski, T.J.: Temporal Difference Learning of Position Evaluation in the Game of Go. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) *Advances in Neural Information Processing Systems*, vol. 6, pp. 817–824. Morgan Kaufmann, San Francisco (1994)
13. Freisleben, B., Luttermann, H.: Learning to Play the Game of Go-Moku: A Neural Network Approach. *Australian Journal of Intelligent Information Processing Systems* 3(2), 52–60 (1996)
14. Gauci, J., Stanley, K.: Generating large-scale neural networks through discovering geometric regularities. In: *GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 997–1004 (2007)
15. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 2673–2681 (1997)
16. Baldi, P., Pollastri, G.: The principled design of large-scale recursive neural network architectures DAG-RNNs and the protein structure prediction problem. *Journal of Machine Learning Research* 4, 575–602 (2003)
17. Graves, A., Schmidhuber, J.: Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In: *NIPS* (2008)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Computation* 9(9), 1735–1780 (1997)
19. Gherman, S.: Atari-Go Applet (2000), <http://www.361points.com/capturego/>
20. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
21. Gomez, F., Miikkulainen, R.: Incremental Evolution of Complex General Behavior. *Adaptive Behavior* 5, 317–342 (1997)

Reservoir Size, Spectral Radius and Connectivity in Static Classification Problems

Luís A. Alexandre¹ and Mark J. Embrechts²

¹ Department of Informatics, University of Beira Interior and
IT - Instituto de Telecomunicações, Covilhã, Portugal

² Rensselaer Polytechnic Institute - Decision Sciences and Engineering Systems
CII 5219, Troy, NY 12180, USA

Abstract. Reservoir computing is a recent paradigm that has proved to be quite effective given the classical difficulty in training recurrent neural networks. An approach to using reservoir recurrent neural networks has been recently proposed for static problems and in this paper we look at the influence of the reservoir size, spectral radius and connectivity on the classification error in these problems. The main conclusion derived from the performed experiments is that only the size of the reservoir is relevant with the spectral radius and the connectivity of the reservoir not affecting the classification performance.

1 Introduction

Recently a new paradigm for Recurrent Neural Networks (RNNs) has been proposed with the Echo State Networks (ESN) [1] and the Liquid State Machines (LSM) [2]. It is called reservoir computation [3].

These types of networks have been used traditionally for time domain tasks. But a proposal for its use in static classification problems has been done in [4][5]. These papers showed how the reservoir approach could be used in static classification problems and made a benchmark study showing that this approach is quite competitive. One of the key issues is the specification of the reservoir: its size, connectivity and spectral radius. These values depend, of course, on the influence they have on the performance of the learning machine. A study on how they influence the performance is needed.

In this paper we perform such study: we made controlled experiments where only one of these parameters is varied at a time such that its effect on the performance can be observed. This is not straightforward since changing either the size, connectivity or the random initialization of the reservoir weights changes the spectral radius in a way that is not analytically foreseeable.

The presented experiments showed the following results: first, the size of the reservoir in terms of the number of neurons it possesses is directly related to the classification performance: as the size increases so does the error. Second, the spectral radius does not seem to have any influence on the performance, as long as it is under 1.0 (otherwise the stabilization process would not converge). This is in contrast to what seems to be the case of time dependent problems [6]. The third conclusion is that the connectivity does not seem to affect the performance either.

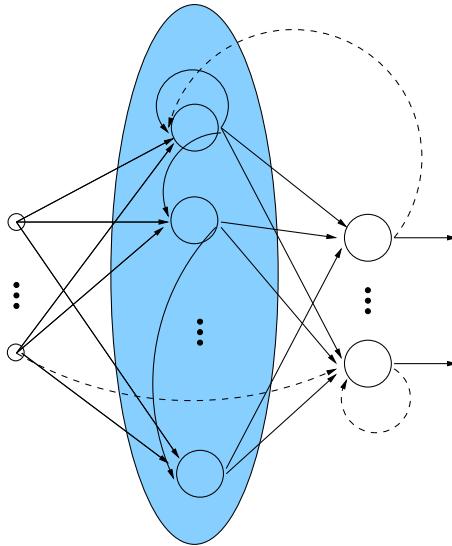


Fig. 1. Representation of a reservoir network. Large circles are the neurons and the small ones represent the input data. The middle (blue) layer is the reservoir. Dashed connections are optional (and not used in this paper). Only a small subset of all possible connections is represented.

The remaining of the paper is organized as follows: the next section presents a brief introduction to reservoir computing; section 3 explains how reservoir computing can be adapted to static classification problems. In section 4 we present the experiments and the last section contains the conclusions.

2 Reservoir Computing

The idea of reservoir computing revolves around using a large recurrent hidden layer (called the reservoir) with fixed weights and only adjusting the output layer weights (see figure 1). Since the output layer neurons use linear activation functions (usually the identity function), learning can be done with a linear system solver. This solves a big problem in traditional RNNs training which had to deal with the minimization of a usually complex function.

For the most simple case (the only recurrent connections are in the reservoir), we can write the state $x(t+1)$ of a N neurons reservoir at time $t+1$, of an ESN with K inputs and L outputs, as:

$$x(t+1) = f(Wx(t) + W^{in}u(t+1)) \quad (1)$$

where $f(\cdot)$ is the reservoir nonlinear activation function, W is the $N \times N$ reservoir weight matrix, W^{in} is the $N \times K$ input weight matrix and $u(t+1)$ is the input data.

The L -dimensional network output is given by

$$y(t) = g(W^{out}x(t)) \quad (2)$$

where $g(\cdot)$ is the output linear activation function and W^{out} is a $L \times N$ output weight matrix. We consider the activation function $g(\cdot)$ to be the identity function.

Regarding the solution of the linear problem that yields the output layer weights, any linear solver can be used. Consider m to be the number of training patterns. The states are collected in a $m \times N$ matrix, S , and the desired values go into a $m \times L$ matrix, D . The system to be solved is

$$S(W^{out})' = D \quad (3)$$

The problem with this formulation is that S is not squared and depends on m . With a simple operation it is possible to deal with both these issues: just multiply both sides by S'

$$(S'S)(W^{out})' = (S'D) \quad (4)$$

This transformation does not change the solution of the system but makes it independent of m and with the new matrix $R = S'S$, square.

Moreover, more evolved ways than the simple solver can be used, such as, ridge regression or partial least squares [7]. The problem with these approaches is that another parameter must be found: in the first case, the regularization parameter and in the second, the number of latent variables. In this paper we used the LAPACK's [8] linear solver to solve directly the system in (4) thus requiring no further parameters but not benefiting from regularization.

3 Reservoir Approach to Static Pattern Classification

The way to use a reservoir network for static pattern classification was introduced, to our best knowledge, in [4].

The issue was that the reservoir had a dependency of its state, $x(t+1)$, on the previous network state, $x(t)$, as can be seen in equation (1).

To break this dependency, that is meaningless in static classification tasks, a stabilization phase was proposed in [4]: keep each input signal present in the network input until the outputs of the reservoir neurons have almost no change. During stabilization the output layer is ignored. This way the reservoir can now process data that is not ordered in time.

The index t in the equations is only used to distinguish between different patterns and does not represent time.

The state equation is iterated (upper index) until $x(t+1)$ does not change significantly. Notice that we do not use the activation function here.

$$x(t+1)^{(i)} = Wx(t+1)^{(i-1)} + W^{in}u(t+1) \quad (5)$$

where $x(t+1)^{(0)} = 0$. Equation (5) replaces equation (1) in the case of static classification tasks.

4 Experiments

In this section we present experiments to evaluate the importance of the reservoir size, its spectral radius and connectivity.

4.1 Datasets

We chose 5 datasets freely available to perform the tests.

All datasets are from the UCI repository [9] with the exception of Olive which is from [10]. Table I contains their main features.

Table 1. Dataset list with its properties

Dataset	N. classes	N. features	N. points
Diabetes	2	8	768
Iris	3	4	150
Olive	9	8	572
Sonar	2	60	208
Wine	3	13	178

4.2 Parameters

Consider N to be the number of neurons in the reservoir, s the connectivity of the neurons in the reservoir (e.g., $s = 0.1$ means that only 10% of the possible connections are not null) and ρ to be the spectral radius (the size of the largest eigenvector) of the reservoir weight matrix.

The issue in these experiments was: how to vary only one of the parameters without varying the remaining? It is not possible to change the spectral radius, ρ , directly. It depends on N , s , the distribution of the random weights in the reservoir and also on the range of possible values for these weights: $[-init, init]$. Changes in either N , s or $init$ will all change ρ in a way that is not possible to analytically predict since the matrices are randomly generated.

To be able to vary only one of the parameters we generated 10 random reservoirs for each triplet $(N, s, init)$ and plotted the average ρ . This was done keeping the value of N fixed. The left plot in figure 2 shows the results when N was kept at 200. The right plot presents the contours of left plot: this information was used to follow a contour and find the values of $init$ that made ρ constant for different values of s , for this particular value of N . The whole process was repeated for all values of N listed above.

A similar analysis was done for the case where s was fixed and N and $init$ varied.

Using the information gathered on the contours lines, we produced 10 reservoirs with the desired ρ up to an error of 0.005, by trial and error, for each combination on N , s and ρ used in the experiments: each reservoir was used in one of the repetitions of a 2-fold cross validation.

In the graphics below, the values of the parameters varied according to:

$$\begin{aligned} N &= \{20, 35, 50, 65, 80, 100, 150, 200, 250, 300, 350\}, \\ s &= \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35\} \text{ and} \\ \rho &= \{0.20, 0.40, 0.60, 0.80, 0.99\}. \end{aligned}$$

4.3 Results

Consider figures 3 to 7. Regarding the left plot, each color ball represents the average error of 10 repetitions of a 2-fold cross validation, on the respective dataset. The color

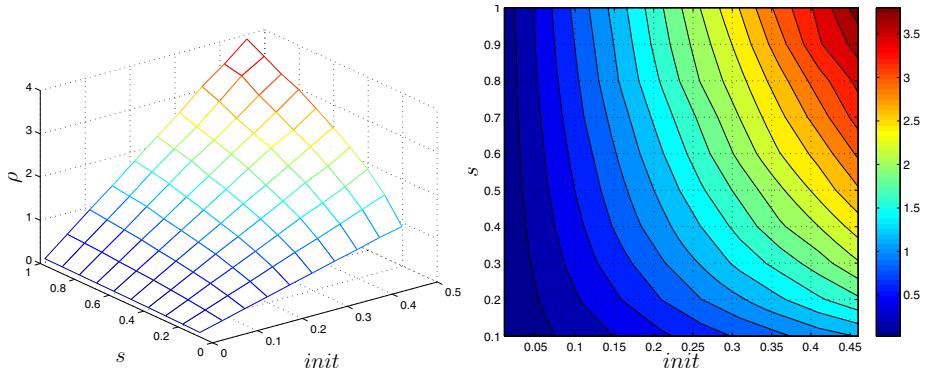


Fig. 2. The left plot shows the spectral radius as a function of connectivity and the half-size of the random reservoir weights $init$. The right figure contains the contour plot of left one. The values of the pairs $(s, init)$ for the first 5 contours from the left were used to keep the spectral radius fixed for varying s .

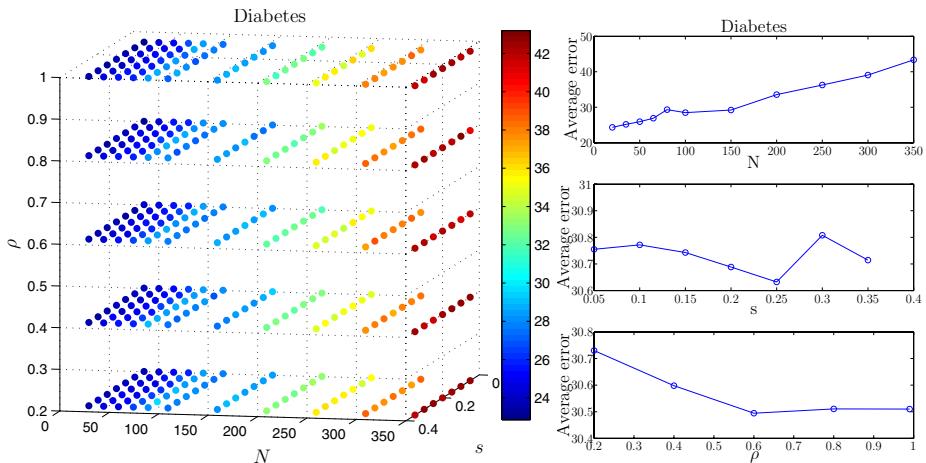
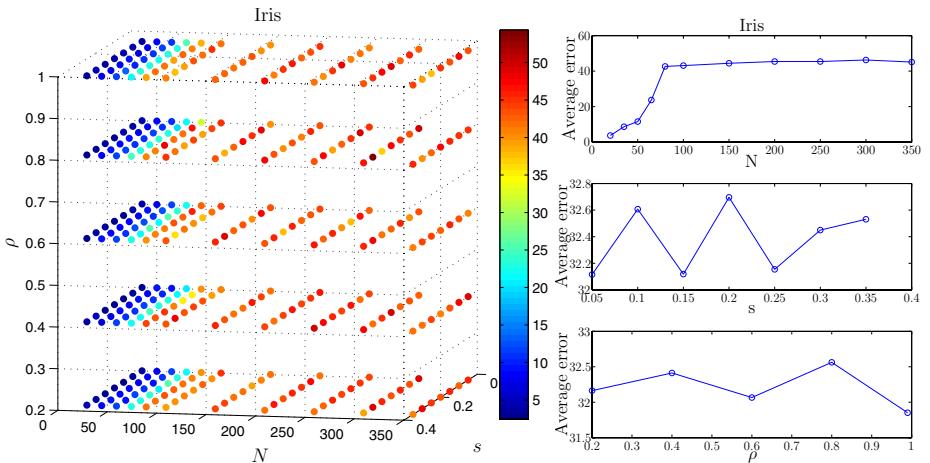
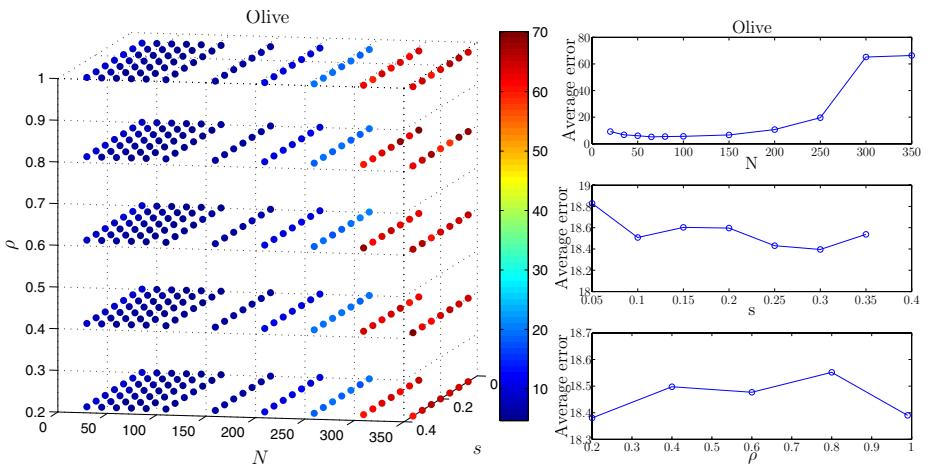


Fig. 3. Results for the Diabetes dataset

encodes the average error according to the scales by the main graphics. The right plots contain the average error for each of N , s and ρ found averaging the error over all the remaining parameters, i.e., for $N = 20$, the figures show the average error when $N = 20$ for all the possible values of s and ρ .

4.4 Condition Number

To evaluate if the problem with increasing the size of the reservoir is due to numeric instability in the linear system solver, we found the condition number of the matrix R for each dataset.

**Fig. 4.** Results for the Iris dataset**Fig. 5.** Results for the Olive dataset

The condition number of a matrix A is $\kappa(A) = \|A\| \cdot \|A^{-1}\|$. If we choose the L2 norm we find that $\kappa(A) = \sigma_{max}/\sigma_{min}$ where σ_{max} is the maximum and σ_{min} the minimum singular values of A . The condition number should be small to ensure the stability of the numeric calculations.

Since the previous experiments showed that the values of the spectral radius and the connectivity did not influence the performance, we repeated the experiments just taking N into account (fixing the other two parameters).

Figure 8 shows the value of the condition number for each of the datasets as N varies.

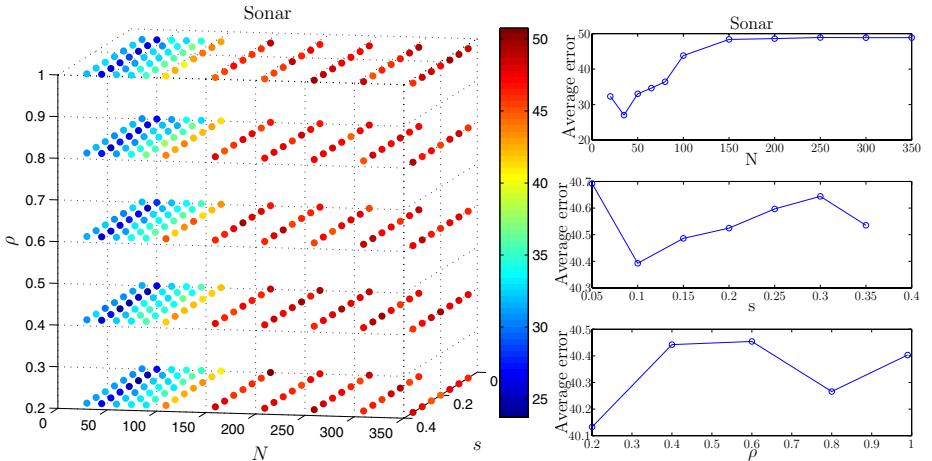


Fig. 6. Results for the Sonar dataset

Table 2. Best results and respective parameters for all datasets

Dataset	Error	Std. dev.	N	s	ρ
Diabetes	22.85	1.07	20	0.05	0.99
Iris	2.47	1.04	20	0.20	0.80
Olive	4.63	0.52	65	0.25	0.80
Sonar	23.75	2.9	35	0.15	0.20
Wine	3.99	0.93	20	0.10	0.20

4.5 Discussion

Table 2 presents the best results for each dataset and the respective parameters. The values do not point to a “best” value of s or ρ . But for N , we see that smaller values are preferred (3 out of 5 of the values are the smallest tested, 20; another is the next in terms of size, 35; the last one is 65 which is still much smaller than the largest tested value of 350).

From the figures 3 to 7 and table 2 we conclude that as N increases the error increases for all datasets, with the exception of Sonar in which case the error decreases from $N = 20$ to $N = 35$ and then continues increasing. The error for the Olive dataset also shows a small decrease for the first values of N but it rapidly increases as N grows. This is surprising as one would suppose no harm would come from an oversized reservoir, other than computational cost. What appears to be happening is a problem with the liner system solving. As can be seen from the values of the condition number of the matrix R in figure 8, the problems become ill-conditioned very fast with the increase in the size N .

The spectral radius and the connectivity do not seem to influence the performance at all. Notice the behavior of error as a function of these two parameters in the two lower right plots of figures 3 to 7 and also the figure’s scale: parameter changes only make

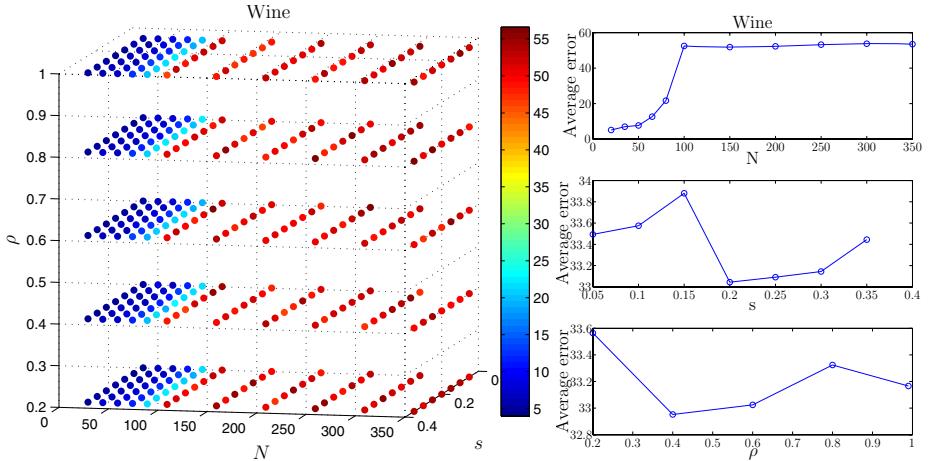


Fig. 7. Results for the Wine dataset

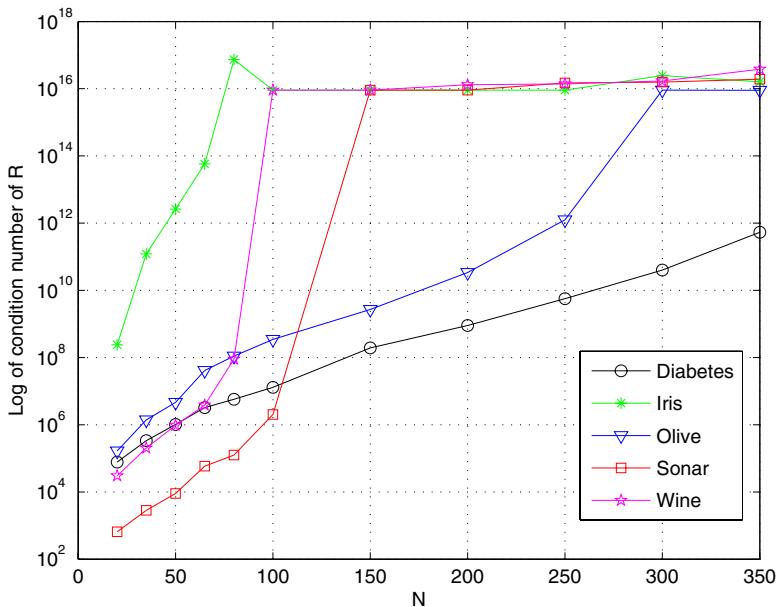


Fig. 8. Logarithm of the condition number of R for each of the datasets as N varies

minor random changes in the performance. This point is made clear in table 3 where the difference between the largest and the smallest values of the error when both the spectral radius and the connectivity are varied, is presented for all datasets, in absolute value and as a percentage of the mean error. The differences in the error that come from varying both the spectral radius and the connectivity are always under 0.9% in absolute terms

Table 3. Percentage changes in the error induced by changes in the spectral radius and the connectivity both in absolute and relative terms

	Spectral radius		Connectivity	
	Absolute [%]	Relative [%]	Absolute [%]	Relative [%]
Diabetes	0.24	0.77	0.18	0.57
Iris	0.71	2.21	0.58	1.79
Olive	0.17	0.93	0.43	2.33
Sonar	0.32	0.80	0.30	0.74
Wine	0.61	1.85	0.84	2.50

and under 2.5% in relative terms. With the exception of the change in the spectral radius for the `Diabetes` dataset that decreases the error as ρ increases (see bottom right plot in figure 3) there is no other linear relation between the changes in these parameters and the error variations. Even in this case, we believe that this apparent inverse correlation is causal since only 5 different values of ρ are considered.

Regarding the values of N at which the values of the error stop increasing (top right plot in figures 3 to 7), they are the same as the ones where the condition number also stops increasing. These values of N are the ones closer to the size of the training sets. Remember, the evaluation is done using 2-fold cross validation. For instance, in the case of `Iris`, the training set will have 75 points. The error shows its peak around this value of N (see the right plot in figure 4) and also the condition number stops increasing after this value (see figure 8). So one concludes that, when using a simple linear solver, the value of N should always be smaller than the size of training set.

5 Conclusions

This paper presented a study on the influence of several parameters on the classification performance of reservoir networks for static pattern classification.

The study showed that: 1) The spectral radius of the reservoir matrix does not influence the classification performance given that it remains in the interval $(0, 1)$ such that the stabilization process converges; 2) An increase in the size of the reservoir increases the error apparently as a result of the ill-conditioning of the linear system solver (improvements might be obtained using, for instance, ridge regression); 3) The connectivity does not change the results; 4) The value of N should not exceed the size of training set when using a simple linear solver.

References

1. Jaeger, H.: ‘The echo state’ approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, Fraunhofer Institute for Autonomous Intelligent Systems (2001)
2. Maas, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14(11), 2531–2560 (2002)

3. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. *Neural Networks* 20, 391–403 (2007)
4. Embrechts, M., Alexandre, L., Linton, J.: Reservoir computing for static pattern recognition. In: 17th European Symposium on Artificial Neural Networks – ESANN 2009, Bruges, Belgium (2009)
5. Alexandre, L., Embrechts, M., Linton, J.: Benchmarking reservoir computing on time-independent classification tasks. In: International Joint Conference on Neural Networks – IJCNN 2009, Atlanta, Georgia, USA, pp. 89–93 (2009)
6. Legenstein, R., Maass, W.: Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks* 20, 323–334 (2007)
7. Wold, S., Sjöström, M., Eriksson, L.: PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems* 58, 109–130 (2001)
8. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, 3rd edn. Society for Industrial and Applied Mathematics, Philadelphia (1999)
9. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998), <http://www.ics.uci.edu/~mlearn/MLRepository.html>
10. Forina, M., Armanino, C.: Eigenvector projection and simplified nonlinear mapping of fatty acid content of italian olive oils. *Ann. Chem.* 72, 125–127 (1981)

Author Index

- Abdel Hady, Mohamed Farouk I-121
Abdullah, Rudwan I-198
Abe, Shigeo I-854
Aida, Shogo I-767
Aihara, Kazuyuki I-306, II-325, II-563, II-573
Aknin, Patrice II-416
Alexandre, Luís A. I-1015
Allinson, Nigel M. II-445
Almeida, Leandro M. I-485
Alonso-Betanzos, Amparo I-824
Alzate, Carlos II-315
Andreou, Panayiotis C. I-874
Antonelo, Eric A. I-747
Antonic, Radovan II-475
Antoniou, Pavlos II-986
Apolloni, Bruno I-449
Araujo, Aluizio F.R. I-515
Arcay, Bernardino II-378
Atiya, Amir F. II-275
Austin, Jim I-728

Ban, Tao I-913
Barbosa, Alexandre Ormiga G. II-495
Barth, Erhardt II-923
Bassani, Hansenclever F. I-515
Bassis, Simone I-449
Bayer, Justin II-755
Belle, Vanya Van I-60
Bertol, Douglas W. II-845
Bhaya, Amit I-668
Blekas, Konstantinos II-145
Bodyanskiy, Yevgeniy I-718
Bogdan, Martin I-181
Borisov, Alexander II-944
Bors, Adrian G. II-245
Bouchain, David I-191
Boutalis, Yiannis II-875
Brdys, Mietek A. II-823
Bumelienė, Skaidra I-618
Byadarhaly, Kiran V. I-296

Cacciola, Matteo II-455
Calcagno, Salvatore II-455
Canals, Vincent I-421

Cardin, Riccardo II-105
Carvajal, Gonzalo I-429
Castellanos Sánchez, Claudio II-745
Cecchi, Guillermo I-587
Čerňanský, Michal I-381
Cernega, Daniela II-865
Chairez, Isaac II-552
Chambers, Jon I-198
Charalambous, Chris I-874
Chen, H. I-401, I-410
Chen, Huanhuan II-185
Chen, Ning II-426
Chenli, Zhang II-345
Cherkassky, Vladimir I-932
Cheung, Yiu-ming I-10
Chinea, Alejandro I-952
Chouchourelou, Arieta I-678
Christodoulidou, K. II-145
Christodoulou, Chris I-737
Christodoulou, Christodoulos S. II-885
Christodoulou, Manolis II-875
Chudacek, Vaclav II-485
Chung, Kwok Wai II-534
Ciccazzo, Angelo I-449
Cleanthous, Aristodemus I-737
Cobb, Stuart I-229
Côme, Etienne II-416
Conti, Marco I-325
Contreras-Lámus, Victor I-277
Corbacho, Fernando II-235
Costa, Ivan G. II-20
Cpalka, Krzysztof II-435
Ćulibrk, Dubravko II-633
Cutsuridis, Vassilis I-229
Cuzzola, Maria II-10

Dafonte, Carlos II-378
Dai, Wuyang I-932
Dal Seno, Bernardo II-515
da Silva, Ivan I-807
Daucé, Emmanuel I-218
Davoian, Kristina I-111
De Brabanter, Jos I-100
De Brabanter, Kris I-100

- Debruyne, Michiel I-100
 De Carvalho, Francisco A.T. I-131
 de Carvalho, Luís Alfredo V. II-65
 de las Rivas, Javier II-195
 de M. Silva Filho, Telmo I-799
 De Moor, Bart I-100, II-315
 Dendek, Cezary I-141
 Denœux, Thierry II-416
 de Paúl, Ivan I-421
 De Pieri, Edson R. II-845
 de Souto, Marcilio C.P. II-20
 de Souza, Renata M.C.R. I-799
 Diaz, David Ronald A. II-495
 Direito, Bruno II-1
 Dolenko, Sergey I-373, II-295, II-397
 Donangelo, Raul II-65
 Dong, Ming II-964
 do Rego, Renata L.M.E. I-515
 Dorronsoro, José R. I-904
 Doulamis, Anastasios II-715, II-895
 Doulamis, Nikolaos II-895
 Dourado, António II-1
 Doya, Kenji I-249
 Drchal, Jan II-775
 Duch, Włodzisław I-151, I-789
 Ďuračková, Daniela I-363
 Dutoit, Pierre I-277
 Eggert, Julian II-804, II-855
 El-Laithy, Karim I-181
 Eliades, Demetrios G. II-954
 Embrechts, Mark II-175, I-1015
 Erwin, Harry I-208
 Espinoza, Marcelo II-315
 Fernandes, João L. I-356
 Fernandez, Emmanuel I-296
 Fiasché, Maurizio II-10
 Figueroa, Miguel I-429
 Filgueiras, Daniel I-515
 Flauzino, Rogerio I-807
 Fontenla-Romero, Oscar I-824
 Foresti, Loris II-933
 Franco, João M. II-1
 Fuentes, Rita II-552
 Galatsanos, Nikolaos I-50, I-942
 García, Rodolfo V. I-259
 García-Sebastián, Maite II-725
 Gastaldo, Paolo II-684
 Gáti, Kristóf I-698
 Gavalda, Arnau I-525
 Ge, Ju Hong II-534
 Georgoulas, George II-485
 Gianniotis, Nikolaos I-567
 Gomez, Faustino J. II-765
 González, Jesús I-259
 Gopych, Petro II-54
 Gori, Marco II-653
 Goryll, Michael II-265
 Gouko, Manabu I-628
 Graham, Bruce P. I-229
 Graña, Manuel II-725
 Grim, Jiří II-165
 Grochowski, Marek I-151
 Gross, Horst-Michael II-735, II-804
 Guan, XuDong II-30
 Guazzini, Andrea I-325
 Guillén, A. II-215
 Guillén, Alberto I-1, II-406
 Gurney, Kevin I-198
 Gutmann, Michael II-623
 Guzhva, Alexander I-373, II-295, II-397
 Hajek, Petr II-505
 Hajnal, Márton A. I-315
 Haker, Martin II-923
 Hans, Alexander I-70
 Hara, Kazuyuki I-171
 Haraguchi, Yuta II-325
 Haralambous, Haris II-335
 Hartley, Matthew II-40
 Hasegawa, Osamu I-864, II-345
 Hasler, Stephan II-663
 Hatakeyama, Takashi I-767
 Hauser, Florian I-191
 He, Shan II-185
 Hellbach, Sven II-804
 Heng, Pheng Ann I-995
 Hernández, Carmen II-725
 Hernández-Lobato, Daniel I-90
 Herrera, Luis Javier II-215, II-406
 Hilbers, Peter A.J. II-305
 Hobson, Stephen I-728
 Höffken, Matthias I-757
 Honda, Hidehito I-678
 Honkela, Timo II-305
 Hora, Jan II-165
 Horváth, Gábor I-698
 Huang, Tingwen II-583

- Huang, Yu II-534
 Hubert, Mia I-100
 Huptych, Michal II-485
 Hussain, Amir I-198
 Hyvärinen, Aapo II-623, II-704
- Iacopino, Pasquale II-10
 Ikeguchi, Tohru I-306, II-325, II-563, II-573
 Iordanidou, Vasiliki II-368
 Ishii, Shin I-608
 Ishikawa, Masumi I-546
 Ito, Koji I-628
- Joshi, Prashant I-239
 Jovic, Mate II-475
- Kadobayashi, Youki I-913
 Kakemoto, Yoshitsugu I-688
 Kanevski, Mikhail II-933
 Kapral, Ondrej II-775
 Kasabov, Nikola II-10, I-913
 Kasai, Wataru I-864
 Kaszkurewicz, Eugenius I-668
 Kato, Hideyuki I-306
 Kato, Yuko II-544
 Kawakami, Hajimu II-673
 Kayastha, Nagendra II-387
 Kemppi, Paul II-954
 Kencana Ramli, Caroline Dewi Puspa II-85
 Kirstein, Stephan II-663
 Kirvaitis, Raimundas I-618
 Kiyokawa, Sachiko I-678
 Kleihorst, Richard I-439
 Knee, Peter II-265
 Kobos, Mateusz II-125
 Koene, Ansgar I-267
 Kolesnik, Marina I-757
 Kollia, Ilianna II-465
 Konnanath, Bharatan II-265
 Körner, Edgar II-663, II-804
 Korytkowski, Marcin I-817
 Koskela, Markus I-495
 Kosmatopoulos, Elias B. I-638
 Kottas, Thodoris II-875
 Kounoudes, Anastasis II-913
 Koutnýk, Jan II-775
 Kovacevic, Asja II-475
 Kovacevic, Darko II-475
- Kryzhanovsky, Vladimir I-844
 Kukolj, Dragan II-633
 Kůrková, Věra I-708
 Kuroe, Yasuaki II-673
 Kuroiwa, Jousuke II-544
 Kurokawa, Hiroaki I-834
 Kwon, Jun Soo I-345
- Laaksonen, Jorma I-495, II-694
 Laganá, Filippo II-455
 Lagaris, I.E. II-145
 Lago-Fernández, Luis F. II-235
 Lang, E.W. II-115
 Laoudias, Christos II-954
 Lawniczak, Anna T. II-358
 Lee, John A. II-95
 Lendasse, Amaury I-1, II-305
 Lhotska, Lenka II-485
 Li, Xiaoli II-185
 Lió, Pietro I-325, II-358
 Likas, Aristidis I-50, I-942, II-205
 Lim, Zhan Wei II-976
 Lindh-Knuutila, Tiina II-305
 Lippe, Wolfram-M. I-111
 Litinskii, Leonid I-161
 Liu, Jindong I-208
 Loengarov, Andreas II-613
 Loizou, Christos P. II-885
 López, Jorge I-904
 Lu, Chih-Cheng I-401, I-410
 Ludermir, Teresa I-30, I-131, I-485, I-557
 Luo, Xuanwen II-964
- Magdi, Yasmine M. I-894
 Maggini, Marco II-653
 Maguire, Liam I-335, II-794
 Mańdziuk, Jacek I-141, II-125
 Manolis, Diamantis I-638
 Manteiga, Minia II-378
 Marakkakis, Apostolos I-942
 Marotta, Angelo I-449
 Martín-Merino, Manuel II-195
 Martínez-Muñoz, Gonzalo I-90
 Martínez-Rego, David I-824
 Martinetz, Thomas II-923
 Martinez, Régis II-75
 Martins da Silva, Rodrigo I-475
 Martins, Nardênia A. II-845
 Martzoukos, Spiros H. I-874

- Maszczuk, Tomasz I-789
 Mathew, Trupthi II-265
 Matsuda, Yoshitatsu II-135
 Matsuka, Toshihiko I-678
 Matsushima, Fumiya I-767
 Matsuura, Takafumi II-563, II-573
 Matsuyama, Yasuo I-767
 Matteucci, Matteo II-515
 Mc Bride, Michael F. II-794
 McGinnity, T.M. I-335, II-794
 Megali, Giuseppe II-455
 Meggiolaro, Marco Antonio II-495
 Meijer, Peter B.L. I-439
 Melacci, Stefano II-653
 Mercier, David II-255
 Mesiano, Cristian I-449
 Michalopoulos, Kostas II-368
 Miche, Yoan II-305
 Michielan, Lisa II-105
 Milis, Marios II-913
 Minai, Ali A. I-296
 Mirikitani, Derrick Takeshi I-975
 Mitsuishi, Takashi I-598
 Miyoshi, Seiji I-171
 Moore, Roger K. II-784
 Morabito, Francesco Carlo II-10, II-455
 Mori, Takeshi I-608
 Moro, Stefano II-105
 Morro, Antoni I-421
 Motohashi, Shun II-563, II-573
 Mourelle, Luiza de Macedo I-475
 Müller, Marco K. II-643
 Murray, Victor II-885
 Mykolaitis, Gytis I-618
- Nagaya, Shigeki II-345
 Nakagawa, Masahiro II-593
 Nakagawa, Masanori I-779
 Nakamura, Yuichi II-593
 Nakano, Ryohei II-155
 Nakano, Takashi I-249
 Nakasuka, Shinchi I-688
 Nakayama, Yoichi I-171
 Nascimento, André C.A. II-20
 Nasios, Nikolaos II-245
 Necoara, Ion II-835
 Nedjah, Nadia I-475
 Ni, He I-577
 Nicolaides, Andrew II-885
 Nishida, Youichi I-767
- Nowicki, Robert I-817
 Ntalianis, Klimis II-895
- Oberhoff, Daniel I-757
 Obornev, Eugeny II-397
 Ochiai, Nimiko I-767
 Odaka, Tomohiro II-544
 Ogura, Hisakazu II-544
 Oja, Erkki I-20, II-305
 Okada, Masato I-171
 Olej, Vladimir II-505
 Ordóñez, Diego II-378
 Osana, Yuko I-505
 Ott, Thomas II-525
 Ouarbya, Lahcen I-975
 Oukhellou, Latifa II-416
- Paechter, B. II-215
 Palm, Günther I-121, I-191, I-894
 Panayiotou, Christos G. II-954
 Pang, Shaoning I-913
 Pantziaris, Marios II-885
 Papadopoulos, Harris II-335
 Papageorgiou, M. I-638
 Parviainen, Elina II-225
 Passarella, Andrea I-325
 Patan, Krzysztof I-80
 Patan, Maciej I-80
 Pattichis, Constantinos S. II-885
 Pattichis, Marios S. II-885
 Paugam-Moisy, Hélène II-75
 Pavlou, Maria II-445
 Pazos, Fernando A. I-668
 Pelckmans, Kristiaan I-60, I-100
 Pellicanó, Diego II-455
 Perdoor, Mithun I-296
 Perez-Gonzalez, David I-208
 Perkiö, Jukka II-704
 Perrig, Stephen I-277
 Persiantsev, Igor I-373, II-295, II-397
 Piekniewski, Filip II-603
 Pitsillides, Andreas II-986
 Pokrić, Maja II-633
 Polycarpou, Marios M. II-954
 Pomares, Héctor II-215, II-406
 Popa, Cosmin Radu I-459
 Popov, Sergiy I-718
 Porto-Díaz, Iago I-824
 Poulsen, Thomas M. II-784
 Pozdnoukhov, Alexei II-933

- Poznyak, Alexander II-552
 Poznyak, Tatyana II-552
 Prasad, Shalini II-265
 Prentis, Philip I-495
 Prescott, Tony J. I-267
 Pribacic, Nikica II-475
 Prieto, Alberto I-259
 Prudêncio, Ricardo B.C. I-30, II-20
 Psillakis, Haris I-648, II-815
- Ralaivola, Liva I-884
 Rallo, Robert I-525
 Ramos, João P. II-1
 Rao, A. Ravishankar I-587
 Raschman, Emil I-363
 Raychaudhury, Somak I-567
 Rebrova, Olga II-435
 Redi, Judith II-684
 Rees, Adrian I-208
 Refaat, Khaled S. II-275
 Ribeiro, Bernardete I-923
 Rigutini, Leonardo I-40
 Rinaudo, Salvatore I-449
 Rizzo, Riccardo I-536
 Rodríguez, Roberto I-259
 Rojas, Fernando I-259
 Rojas, Ignacio I-1, II-215, II-406
 Rosselló, Josep L. I-421
 Rubio, Ginés I-1, II-215, II-406
 Ruiz-Llata, Marta I-467
 Runger, George II-944
 Rutkowski, Leszek II-435
- Sakkalis, Vangelis II-368
 Sánchez-Montañés, Manuel II-235
 Sánchez Orellana, Pedro L. II-745
 San Román, Belén I-259
 Santos, Jorge M. II-175
 Särelä, Jaakko II-285
 Sattigeri, Prasanna II-265
 Schaffernicht, Erik II-735
 Schaul, Tom I-1005
 Scherer, Rafal I-817
 Scherer, Stefan I-894
 Schmidhuber, Jürgen I-964, I-1005,
 II-755, II-765
 Schrauwen, Benjamin I-747, I-985
 Schüle, Martin II-525
 Schwenker, Friedhelm I-121, I-894
 Seah, Winston Khoon-Guan II-976
- Shaposhnyk, Vladyslav I-277
 Shibata, Tadashi I-391
 Shidama, Yasunari I-598
 Shimada, Yutaka II-325
 Shimelevich, Mikhail II-397
 Shirai, Haruhiko II-544
 Shirotori, Tomonori I-505
 Shrestha, Durga Lal II-387
 Shugai, Julia II-295
 Silva, Catarina I-923
 Simou, Nikolaos II-465
 Sittiprapaporn, Wichian I-345
 Sjöberg, Mats I-495
 Šnorek, Miroslav II-775
 Soares, Rodrigo G.F. I-131
 Solanas, Agusti I-525
 Solea, Razvan II-865
 Solomatine, Dimitri P. II-387
 Sorjamaa, Antti I-1
 Souza, Jefferson R. I-485
 Spaanenburg, Lambert I-439
 Spanias, Andreas II-265
 Sperduti, Alessandro II-105
 Sportiello, Luigi II-515
 Spreckley, Steve I-567
 Stafylopatis, Andreas I-942, II-465
 Stamou, Giorgos II-465
 Stempfel, Guillaume I-884
 Stephan, Volker II-735
 Stoop, Ruedi I-618, II-525
 Stylios, Chrysostomos II-485
 Suárez, Alberto I-90
 Suard, Frédéric II-255
 Suykens, Johan A.K. I-60, I-100, II-315
 Suzuki, Takayuki II-573
- Takahashi, Norihiro I-391
 Tamaševičius, Arūnas I-618
 Tamaševičiūtė, Elena I-618
 Tan, Colin Keng-Yan II-976
 Tan, Guan-Hong II-976
 Tanahashi, Yusuke II-155
 Tanscheit, Ricardo II-495
 Taylor, John G. II-40
 Tehrani, Mona Akbarniai I-439
 Teixeira, Ana R. II-115
 Terai, Asuka I-779
 Tereshko, Valery II-613
 Theodosiou, Zenonas II-905, II-913
 Thornton, Trevor II-265

- Tiño, Peter I-567
 Titov, Mykola I-718
 Tobe, Yutaro I-864
 Togelius, Julian II-755, II-765
 Tomé, Ana Maria II-115
 Tomi, Naoki I-628
 Torreão, José R.A. I-356
 Toshiyuki, Hamada II-544
 Tran, Vu Nam II-823
 Trentin, Edmondo I-40
 Triesch, Jochen I-239
 Tsaih, Rua-Huan I-658
 Tsapatsoulis, Nicolas II-895, II-905, II-913
 Tuia, Devis II-933
 Tuv, Eugene II-944
 Tzikas, Dimitris I-50
 Tzortzis, Grigoris II-205
 Udluft, Steffen I-70
 Ueno, Kosuke I-546
 Unkelbach, Jan I-964
 Urso, Alfonso I-536
 Valderrama-Gonzalez, German D. I-335
 Valença, Ivna I-557
 Valenzuela, Olga I-259
 Valenzuela, Waldo I-429
 Valpola, Harri II-285
 van Heeswijk, Mark II-305
 Van Huffel, Sabine I-60
 Vasa, Suresh I-296
 Vasiljević, Petar II-633
 Vassiliades, Vassilis I-737
 Vehtari, Aki II-225
 Velázquez, Luis I-259
 Vellasco, Marley Maria B.R. II-495
 Ventura, André II-1
 Verleysen, Michel II-95
 Verma, Anju II-10
 Versaci, Mario II-455
 Verstraeten, David I-985
 Victer, Silvia M.C. I-356
 Viitaniemi, Ville II-694
 Villa, Alessandro E.P. I-277
 Wan, Wang-Gen II-426
 Wan, Yat-wah I-658
 Wang, Jun II-583
 Wedemann, Roseli S. II-65
 Wei, Hui II-30
 Wermter, Stefan I-208
 Wersing, Heiko II-663
 Wickens, Jeff I-249
 Wierstra, Daan II-755
 Wu, QingXiang I-335
 Würtz, Rolf P. II-643
 Xie, Shengkun II-358
 Xu, Jian II-534
 Yamaguchi, Kazunori II-135
 Yang, Zhirong I-20
 Yao, Xin II-185
 Yébenes-Calvino, Mar I-467
 Yi, Sun I-964
 Yi, Zhang I-287, I-995
 Yin, Hujun I-577
 Yli-Krekola, Antti II-285
 Yonekawa, Masato I-834
 Yoshimoto, Junichiro I-249
 Yu, Jiali I-287
 Zeng, Hong I-10
 Zeng, Zhigang II-583
 Zervakis, Michalis II-368
 Zhang, Chen II-855
 Zhang, Haixian I-287
 Zhang, Lei I-995
 Zlokolica, Vladimir II-633
 Zunino, Rodolfo II-684