Valeri Mladenov Petia Koprinkova-Hristova
Günther Palm Alessandro E.P. Villa
Bruno Appollini Nikola Kasabov (Eds.)

LNCS 8131

Artificial Neural Networks and Machine Learning – ICANN 2013

23rd International Conference on Artificial Neural Networks
Sofia, Bulgaria, September 2013
Proceedings

Springer

Lecture Notes in Computer Science 8131

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
TU Dortmund University, 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 for Informatics, Saarbruecken, Germany

Valeri Mladenov Petia Koprinkova-Hristova
Günther Palm Alessandro E.P. Villa
Bruno Appollini Nikola Kasabov (Eds.)

Artificial Neural Networks and Machine Learning – ICANN 2013

23rd International Conference
on Artificial Neural Networks
Sofia, Bulgaria, September 10-13, 2013
Proceedings

Springer

Volume Editors

Valeri Mladenov
Technical University of Sofia
1000 Sofia, Bulgaria
E-mail: valerim@tu-sofia.bg

Petia Koprinkova-Hristova
Bulgarian Academy of Sciences
1113 Sofia, Bulgaria
E-mail: pkoprinkova@bas.bg

Günther Palm
University of Ulm
89075 Ulm, Germany
E-mail: guenther.palm@uni-ulm.de

Alessandro E.P. Villa
Université de Lausanne
1015 Lausanne, Switzerland
E-mail: alessandro.villa@unil.ch

Bruno Appollini
University of Milano
20135 Milano, Italy
E-mail: apolloni@di.unimi.it

Nikola Kasabov
Auckland University of Technology
Auckland 1010, New Zealand
E-mail: nkasabov@aut.ac.nz

ISSN 0302-9743
e-ISSN 1611-3349
ISBN 978-3-642-40727-7
e-ISBN 978-3-642-40728-4
DOI 10.1007/978-3-642-40728-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013946649

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

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

© Springer-Verlag Berlin Heidelberg 2013
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in ist current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

These proceedings comprise all accepted papers presented during the International Conference on Artificial Neural Networks (ICANN) 2013. The conference is the annual event of the European Neural Network Society (ENNS) that covers all topics in the area of neural networks and their applications. Its scope is wide, ranging from machine learning algorithms to models of real nervous systems. The main goal of the conference is to bring together and to facilitate contacts between researchers from information sciences and neurosciences and to provide a high-level international forum for both the academic and the industrial communities. The conference aims to address new challenges, share solutions, and discuss future research directions in understanding neural and cognitive processes in the brain and in the development of novel artificial neural networks, learning systems, computational intelligence, and real-world intelligent artificial system applications.

The ICANN conferences were founded in 1991 and through the years were established as the flagship conferences of the ENNS. The organizers of the 23rd issue of ICANN are three main Bulgarian scientific institutions, namely, the Technical University of Sofia, the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences, and the Union of Automatic and Informatics. The conference was held during September 10–13, 2013, in the Technical University of Sofia.

During the 23rd ICANN, six plenary talks were given by some of the most internationally established researchers in the neural network community. This year emphasis was also put on the recently announced mega-project of the EU – the Human Brain Project (HBP) with a plenary talk given by Prof. Karlheinz Meier, – a co-director of the project.

Stephen Grossberg, Professor of Cognitive and Neural Systems, Professor of Mathematics, Psychology, and Biomedical Engineering, Director of the Center for Adaptive Systems, Boston University, was another eminent plenary speaker who traditionally attends ICANN events. A plenary talk was also given by the Honorary Chair of ICANN 2013, Prof. Nikola Kasabov, who is Foundation Director of KEDRI and a Chair of Knowledge Engineering at the School of Computer and Information Sciences at AUT, Fellow of the Royal Society of New Zealand, Fellow of IEEE and past president of the INNS. The past president of the ENNS (1999–2001–2004), Prof. Erkki Oja, was also a plenary speaker. He is Professor of Computer Science and Engineering at Aalto University, School of Science and Technology, Director of the Computational Inference (COIN) Research Centre at Aalto and Past Chairman of the Research Council for Natural Sciences and Engineering of the Academy of Finland. The current president of the ENNS Executive Committee, Alessandro E.P. Villa, Director of the Laboratory of Neuroheuristics and Neuroeconomics at the University of Lausanne, was also among the distinguished plenary speakers. A plenary talk was also given by Prof. Günther Palm – an active member of the ENNS executive board. He is Professor in Computer Science and Director of the Institute of Neural Information Processing at the University of Ulm.

The total number of submitted papers to ICANN 2013 was 128. After a thorough peer-review process, the Program Co-chairs selected 78 papers. All papers passed several steps of a review process before being finally accepted The selected papers were divided by subject into the following main topics: neural network theory and models; machine learning and learning algorithms; brain–machine interaction and bio-inspired systems; cognitive science and neuroscience; pattern recognition and classification; applications in control, robotics, natural language processing, and neuro-economics etc. Along with the regular and the poster sessions, two tutorials were organized – by Prof. Bruno Apolloni from the Department of Computer Science at the University of Milan, and by Prof. Alexander Gegov from the School of Computing at Portsmouth University.

Presenting the recent results in neural network theory and important applications, the collected volume will be of interest to all researchers and postgraduate students in the area of computational intelligence, applied mathematics, engineering, neuroscience, and other related areas.

We would like to thank all the participants for their contribution to the conference program and for their contribution to these proceedings. Many thanks go to the Bulgarian organizers for their support and hospitality, which allowed all foreign participants to feel at home. Our special thanks go to our colleague Prof. Nikola Kasabov, who was the Honorary Chair of this conference and helped with its organization. We also express our sincere thanks to all reviewers for their help in the review procedures and their valuable comments and recommendations to ensure the high quality of all contributions in this volume.

July 2013

Valeri Mladenov
Guenther Palm
Alessandro Villa
Bruno Apolloni
Petia Koprinkova-Hristova
Nikola Kasabov

Organization

Honorary Chair

Nikola Kasabov	Auckland University of Technology, New Zealand

General Co-chairs

Valeri Mladenov	Technical University of Sofia, Bulgaria
Vassil Sgurev	Bulgarian Academy of Sciences, Sofia, Bulgaria

Program Co-chairs

Guenther Palm	Universtiyt of Ulm, Germany
Alessandro Villa	UNIL, Switzerland
Bruno Apolloni	University of Milano, Italy
Petia Koprinkova-Hristova	Bulgarian Academy of Sciences, Sofia, Bulgaria
Mincho Hadjiski	Bulgarian Academy of Sciences, Sofia, Bulgaria

Program Committee and Reviewers

Luís Alexandre	UBI - Univ. Beira Interior, Portugal
Mauricio Alvarez	Universidad Tecnológica de Pereira, Colombia
Yoshiyuki Asai	Okinawa Institute of Science and Technology Graduate University, Japan
Lubica Benuskova	University of Otago, New Zealand
Ulysses Bernardet	Simon Fraser University, Vancouver, Canada
Ivo Bukovsky	Czech Technical University in Prague, Czech Republic
Francesco Camastra	University of Naples Parthenope, Italy
Angelo Cangelosi	Plymouth University, UK
Ke Chen	University of Manchester, UK
Jorg Conradt	TU München, Germany
Alessandro Di Nuovo	Plymouth University, UK
Jan Drugowitsch	Ecole Normale Superieure, France
Lan Du	Macquarie University, Sydney, Australia
Péter Érdi	Kalamazoo College, USA
Pablo Estevez	University of Chile, Chile
Igor Farkaš	Comenius University in Bratislava, Slovakia
Mauro Gaggero	National Research Council of Italy, Italy
Petia Georgieva	University of Aveiro, Portugal
Tobias Glasmachers	Ruhr-Universität Bochum, Germany

Giorgio Gnecco	University of Genova, Italy
Andre Gruning	University of Surrey, UK
Barbara Hammer	TU Clausthal, Germany
Stefan Heinrich	Universität Hamburg, Germany
Tom Heskes	Radboud University Nijmegen, The Netherlands
Timo Honkela	Aalto University, Finland
Amir Hussain	University of Stirling, UK
Lazaros Iliadis	Democritus University of Thrace, Greece
Yaochu Jin	University of Surrey, UK
Juha Karhunen	Aalto University, Finland
Dietrich Klakow	University of Saarland, Germany
Mario Koeppen	Kyushu Institute of Technology, Japan
Stefanos Kollias	National Technical University of Athens, Greece
Irena Koprinska	University of Sydney, Australia
Vera Kurkova	Institute of Computer Science, Academy of Sciences of the Czech Republic
Giancarlo La Camera	SUNY Stony Brook, NY, USA
Diego Liberati	Politecnico di Milano, Italy
Aristidis Likas	University of Ioannina, Greece
Teresa Ludermir	Federal University of Pernambuco, Brazil
Mantas Lukoševičius	Jacobs University, Germany
Zhiyuan Luo	Royal Holloway, University of London, UK
Francesco Marcelloni	Universita di Pisa, Italy
Thomas Martinetz	University of Luebeck, Germany
Francesco Masulli	University of Genova, Italy
George Mengov	Sofia University, Bulgaria
Bjoern Menze	ETH Zurich, Switzerland
Alessio Micheli	Universita di Pisa, Italy
Kazushi Mimura	Hiroshima City University
Claudio Mirasso	Institute for Cross-Disciplinary Physics and Complex Systems
Tetsuro Morimura	IBM, Japan
Shinichi Nakajima	Nikon Corporation, Japan
Heiko Neumann	University of Ulm, Germany
Jose Nunez-Yanez	University of Bristol, UK
Sebastian Pannasch	TU Dresden, Germany
Charis Papadopoulos	University of Ioannina, Greece
Jaakko Peltonen	Aalto University, Finland
Tonatiuh Pena Centeno	Universität Greifswald, Germany
Alfredo Petrosino	Università degli Studi di Napoli, Italy
Vincenzo Piuri	University of Milan, Italy
Novi Quadrianto	University of Cambridge, UK
Marina Resta	University of Genova, Italy
Achim Rettinger	Karlsruhe Institute of Technology, Germany

Fabrice Rossi	Université Paris 1 Panthéon-Sorbonne, France
Manuel Roveri	Politecnico di Milano, Italy
Alessandro Rozza	Università degli Studi di Napoli, Italy
Marcello Sanguineti	University of Genova, Italy
Jorge Santos	Instituto Superior de Engenharia do Porto, Portugal
Sohan Seth	Helsinki Institute for Information Technology, Finland
Hiroshi Shimodaira	University of Edinburgh, UK
Alessandro Sperduti	University of Padova, Italy
Johan Suykens	KU Leuven, Belgium
Athanasios Tsadiras	Aristotle University of Thessaloniki, Greece
Giorgio Valentini	University of Milan, Italy
Eleni Vasilaki	University of Sheffield, UK
Vassilios Verykios	Hellenic Open University, Greece
Carmen Vidaurre	Berlin Institute of Technology, Germany
Nathalie Villa-Vialaneix	Université Paris 1, France
Shinji Watanabe	MERL, Cambridge, MA, USA
Roseli Wedemann	Universidade do Estado do Rio de Janeiro, Brazil
Thomas Wennekers	Plymouth University, UK
Stefan Wermter	University of Hamburg, Germany
Heiko Wersing	Honda Research Institute Europe, Germany
Zhirong Yang	Aalto University, Finland
Shanfeng Zhu	Fudan University, China

Additional Reviewers

Angelo Alessandri	University of Genova, Italy
Davide Bacciu	Università di Pisa, Italy
Konstantinos Blekas	University of Ioannina, Greece
Jérémie Cabessa	UNIL, Switzerland
Claudio Ceruti	Università degli Studi di Milano, Italy
Vasileios Chasanis	University of Ioannina, Greece
Angelo Ciaramella	University of Naples “Parthenope”, Italy
Alex Cope	Sheffield University, UK
Federico Corradi	University of Zurich and ETH Zurich, Switzerland
Marco Frasca	University of Milan, Italy
Samuele Grillo	Politecnico di Milano, Italy
Guillaume Hennequin	University of Cambridge, United Kingdom
Ziyuan Lin	Aalto University, Finland
Danilo Macciò	Istituto di Studi su Sistemi Intelligenti per l’Automazione, Bari, Italy
Mario Manzo	University of Naples “Parthenope”, Italy

Michael Pfeiffer	University of Zurich and ETH Zurich, Switzerland
Elisabetta Punta	CNR, Italy
Matteo Re	Università degli Studi di Milano, Italy
Denis Sheynikhovich	University Pierre & Marie Curie, France
Antonino Staiano	University of Naples Parthenope, Italy
Peter Tino	University of Birmingham, United Kingdom
Lorenzo Valerio	CNR, Italy

Local Organizing Committee

Yancho Todorov	Bulgarian Academy of Sciences, Sofia, Bulgaria
Georgi Tsenov	Technical University of Sofia, Bulgaria
Agata Manolova	Technical University of Sofia, Bulgaria
Stanislav Panev	Technical University of Sofia, Bulgaria
Svetlin Antonov	Technical University of Sofia, Bulgaria

Table of Contents

Neural Network Theory and Models

Hessian Corrected Input Noise Models 1
Botond Attila Bócsi and Lehel Csató

Model-Based Clustering of Temporal Data 9
Hani El Assaad, Allou Samé, Gérard Govaert, and Patrice Aknin

Fast Approximation Method for Gaussian Process Regression Using Hash Function for Non-uniformly Distributed Data 17
Yuya Okadome, Yutaka Nakamura, Yumi Shikauchi, Shin Ishii, and Hiroshi Ishiguro

An Analytical Approach to Single Node Delay-Coupled Reservoir Computing 26
Johannes Schumacher, Hazem Toutounji, and Gordon Pipa

Applying General-Purpose Data Reduction Techniques for Fast Time Series Classification 34
Stefanos Ougiaroglou, Leonidas Karamitopoulos, Christos Tatoglou, Georgios Evangelidis, and Dimitris A. Dervos

Two-Layer Vector Perceptron 42
Vladimir Kryzhanovsky, Irina Zhelavskaya, and Juan Antonio Clares Tomas

Local Detection of Communities by Neural-Network Dynamics 50
Hiroshi Okamoto

The Super-Turing Computational Power of Interactive Evolving Recurrent Neural Networks 58
Jérémie Cabessa and Alessandro E.P. Villa

Group Fused Lasso 66
Carlos M. Alaíz, Álvaro Barbero, and José R. Dorronsoro

Exponential Synchronization of a Class of RNNs with Discrete and Distributed Delays 74
Farouk Chérif, Hajer Brahmi, Boudour Ammar, and Adel M. Alimi

Variational Foundations of Online Backpropagation 82
Salvatore Frandina, Marco Gori, Marco Lippi, Marco Maggini, and Stefano Melacci

GNMF with Newton-Based Methods 90
Rafał Zdunek, Anh-Huy Phan, and Andrzej Cichocki

Improving the Associative Rule Chaining Architecture 98
Nathan Burles, Simon O'Keefe, and James Austin

Machine Learning and Learning Algorithms

A Two-Stage Pretraining Algorithm for Deep Boltzmann Machines 106
KyungHyun Cho, Tapani Raiko, Alexander Ilin, and Juha Karhunen

A Low-Energy Implementation of Finite Automata by Optimal-Size Neural Nets 114
Jiří Šíma

A Distributed Learning Algorithm Based on Frontier Vector Quantization and Information Theory 122
Diego Peteiro-Barral and Bertha Guijarro-Berdiñas

Efficient Baseline-Free Sampling in Parameter Exploring Policy Gradients: Super Symmetric PGPE 130
Frank Sehnke

Direct Method for Training Feed-Forward Neural Networks Using Batch Extended Kalman Filter for Multi-Step-Ahead Predictions 138
Artem Chernodub

Learning with Hard Constraints 146
Giorgio Gnecco, Marco Gori, Stefano Melacci, and Marcello Sanguineti

Bidirectional Activation-based Neural Network Learning Algorithm 154
Igor Farkaš and Kristína Rebrová

A Neural Network Model for Online Multi-Task Multi-Label Pattern Recognition 162
Daisuke Higuchi and Seiichi Ozawa

Novel Feature Selection and Kernel-Based Value Approximation Method for Reinforcement Learning 170
Hunor Sandor Jakab and Lehel Csató

Learning of Lateral Interactions for Perceptual Grouping Employing Information Gain 178
Martin Meier, Robert Haschke, and Helge J. Ritter

On–Line Laplacian One–Class Support Vector Machines 186
Salvatore Frandina, Marco Lippi, Marco Maggini, and Stefano Melacci

OSA: One-Class Recursive SVM Algorithm with Negative Samples for Fault Detection 194
Mikhail Suvorov, Sergey Ivliev, Garegin Markarian, Denis Kolev, Dmitry Zvikhachevskiy, and Plamen Angelov

Brain-Machine Interaction and Bio-inspired Systems

EEG Dataset Reduction and Classification Using Wave Atom Transform 208
Ignas Martisius, Darius Birvinskas, Robertas Damasevicius, and Vacius Jusas

Embodied Language Understanding with a Multiple Timescale Recurrent Neural Network 216
Stefan Heinrich, Cornelius Weber, and Stefan Wermter

Unsupervised Online Calibration of a c-VEP Brain-Computer Interface (BCI) 224
Martin Spüler, Wolfgang Rosenstiel, and Martin Bogdan

A Biologically Inspired Model for the Detection of External and Internal Head Motions 232
Stephan Tschechne, Georg Layher, and Heiko Neumann

Cortically Inspired Sensor Fusion Network for Mobile Robot Heading Estimation 240
Cristian Axenie and Jörg Conradt

Learning Sensorimotor Transformations with Dynamic Neural Fields ... 248
Yulia Sandamirskaya and Jörg Conradt

Cognitive Sciences and Neuroscience

Learning Temporally Precise Spiking Patterns through Reward Modulated Spike-Timing-Dependent Plasticity 256
Brian Gardner and André Grüning

Memory Trace in Spiking Neural Networks 264
Marta Castellano and Gordon Pipa

Attention-Gated Reinforcement Learning in Neural Networks—A Unified View 272
Tobias Brosch, Friedhelm Schwenker, and Heiko Neumann

Dynamic Memory for Robot Control Using Delay-Based Coincidence Detection Neurones 280
Francis Jeanson and Tony White

Robust Principal Component Analysis for Brain Imaging 288
Petia Georgieva and Fernando De la Torre

Phase Control of Coupled Neuron Oscillators 296
Mayumi Irifune and Robert H. Fujii

Dendritic Computations in a Rall Model with Strong Distal Stimulation 304
Youwei Zheng and Lars Schwabe

Modeling Action Verb Semantics Using Motion Tracking 312
Timo Honkela and Klaus Förger

Evolution of Dendritic Morphologies Using Deterministic and Nondeterministic Genotype to Phenotype Mapping 319
Parimala Alva, Giseli de Sousa, Ben Torben-Nielsen, Reinoud Maex, Rod Adams, Neil Davey, and Volker Steuber

Sparseness Controls the Receptive Field Characteristics of V4 Neurons: Generation of Curvature Selectivity in V4 327
Yasuhiro Hatori, Tatsuroh Mashita, and Ko Sakai

Pattern Recognition and Classification

Handwritten Digit Recognition with Pattern Transformations and Neural Network Averaging 335
Juan Manuel Alonso-Weber, M. Paz Sesmero, German Gutierrez, Agapito Ledezma, and Araceli Sanchis

Echo State Networks in Dynamic Data Clustering 343
Petia Koprinkova-Hristova and Kiril Alexiev

Self-Organization in Parallel Coordinates 351
Marjan Trutschl, Phillip C.S.R. Kilgore, and Urška Cvek

A General Image Representation Scheme and Its Improvement for Image Analysis 359
Hui Wei, Qingsong Zuo, and Bo Lang

Learning Features for Activity Recognition with Shift-Invariant Sparse Coding 367
Christian Vollmer, Horst-Michael Gross, and Julian P. Eggert

Hearing Aid Classification Based on Audiology Data 375
Christo Panchev, Muhammad Naveed Anwar, and Michael Oakes

BLSTM-RNN Based 3D Gesture Classification 381
Grégoire Lefebvre, Samuel Berlemont, Franck Mamalet, and Christophe Garcia

Feature Selection for Neural Network-Based Interval Forecasting of Electricity Demand Data 389
Mashud Rana, Irena Koprinska, and Abbas Khosravi

A Combination of Hand-Crafted and Hierarchical High-Level Learnt Feature Extraction for Music Genre Classification 397
Julien Martel, Toru Nakashika, Christophe Garcia, and Khalid Idrissi

Exploration of Loneliness Questionnaires Using the Self-Organising Map 405
Krista Lagus, Juho Saari, Ilari T. Nieminen, and Timo Honkela

An Effective Dynamic Gesture Recognition System Based on the Feature Vector Reduction for SURF and LCS 412
Pablo V.A. Barros, Nestor T.M. Júnior, Juvenal M.M. Bisneto, Bruno J.T. Fernandes, Byron L.D. Bezerra, and Sérgio M.M. Fernandes

Feature Weighting by Maximum Distance Minimization 420
Jens Hocke and Thomas Martinetz

Training Computationally Efficient Smartphone–Based Human Activity Recognition Models 426
Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz

A Novel Procedure for Training L1-L2 Support Vector Machine Classifiers 434
Davide Anguita, Alessandro Ghio, Luca Oneto, Jorge Luis Reyes-Ortiz, and Sandro Ridella

Online Classification of Eye Tracking Data for Automated Analysis of Traffic Hazard Perception 442
Enkelejda Tafaj, Thomas C. Kübler, Gjergji Kasneci, Wolfgang Rosenstiel, and Martin Bogdan

Neural Network Applications in Control and Robotics

Time-Series Forecasting of Indoor Temperature Using Pre-trained Deep Neural Networks 451
Pablo Romeu, Francisco Zamora-Martínez, Paloma Botella-Rocamora, and Juan Pardo

Recurrent Fuzzy-Neural Network with Fast Learning Algorithm for Predictive Control 459
Yancho Todorov, Margarita Terzyiska, and Michail Petrov

Real-Time Interface Board for Closed-Loop Robotic Tasks on the SpiNNaker Neural Computing System 467
Christian Denk, Francisco Llobet-Blandino, Francesco Galluppi, Luis A. Plana, Steve Furber, and Jörg Conradt

A Software Framework for Cognition, Embodiment, Dynamics, and Autonomy in Robotics: *cedar* 475
Oliver Lomp, Stephan Klaus Ulrich Zibner, Mathis Richter, Iñaki Rañó, and Gregor Schöner

Adaptive Critic Neural Network Solution of Optimal Control Problems with Discrete Time Delays 483
Tibor Kmet and Maria Kmetova

Emotion Generation System Considering Complex Emotion Based on MaC Model with Neural Networks 495
Tsubasa Takamatsu and Yuko Osana

Neuro-Optimal Controller for Robot Manipulators 503
Mohammed Boukens and Abdelkrim Boukabou

Learning to Walk Using a Recurrent Neural Network with Time Delay 511
Boudour Ammar, Naima Chouikhi, Adel M. Alimi, Farouk Chérif, Nasser Rezzoug, and Philippe Gorce

The Imbalance Network and Incremental Evolution for Mobile Robot Nervous System Design 519
Paul Olivier and Juan Manuel Moreno Arostegui

Balancing of a Simulated Inverted Pendulum Using the NeuraBase Network Model 527
Robert Hercus, Kit-Yee Wong, and Kim-Fong Ho

Coordinated Rule Acquisition of Decision Making on Supply Chain by Exploitation-Oriented Reinforcement Learning -Beer Game as an Example- 537
Fumiaki Saitoh and Akihide Utani

Other Applications of Neural Networks

Using Exponential Kernel for Word Sense Disambiguation 545
Tinghua Wang, Junyang Rao, and Dongyan Zhao

Independent Component Analysis Filtration for Value at Risk Modelling 553
Ryszard Szupiluk, Piotr Wojewnik, and Tomasz Ząbkowski

Wind Power Resource Estimation with Deep Neural Networks 563
Frank Sehnke, Achim Strunk, Martin Felder, Joris Brombach, Anton Kaifel, and Jon Meis

Wavelet Neural Networks for Electricity Load Forecasting – Dealing with Border Distortion and Shift Invariance 571
Mashud Rana and Irena Koprinska

Interactive Two-Level WEBSOM for Organizational Exploration 579
Timo Honkela and Michael Knapek

Optimal Operation of Electric Power Production System without Transmission Losses Using Artificial Neural Networks Based on Augmented Lagrange Multiplier Method 586
George J. Tsekouras, Fotis D. Kanellos, Nikos E. Mastorakis, and Valeri Mladenov

An Echo State Network with Working Memories for Probabilistic Language Modeling 595
Yukinori Homma and Masafumi Hagiwara

Using the Analytic Feature Framework for the Detection of Occluded Objects 603
Marvin Struwe, Stephan Hasler, and Ute Bauer-Wersing

Boltzmann Machines for Image Denoising 611
KyungHyun Cho

Comparison on Late Fusion Methods of Low Level Features for Content Based Image Retrieval 619
Nikolay N. Neshov

Vehicle Plate Recognition Using Improved Neocognitron Neural Network 628
Dmitry Kangin, George Kolev, and Plamen Angelov

Author Index 641

Hessian Corrected Input Noise Models

Botond Attila Bócsi and Lehel Csató

Faculty of Mathematics and Informatics, Babeş-Bolyai University
{bboti,lehel.csato}@cs.ubbcluj.ro

Abstract. When the inputs of a regression problem are corrupted with noise, integrating out the noise process leads to biased estimates. We present a method that corrects the bias caused by the integration. The correction is proportional to the Hessian of the learned model and to the variance of the input noise. The method works for arbitrary regression models, the only requirement is two times differentiability of the respective model. The conducted experiments suggest that significant improvement can be gained using the proposed method. Nevertheless, experiments on high dimensional data highlight the limitations of the algorithm.

1 Introduction

In regression problems we find the optimal mapping that explains the relationship between an input space and an output space. Along with the class from where the model is taken, regression methods assume the presence of noise in the data. Most regression methods assume output noise but neglect the possibility of the input noise due to analytical intractability or the fact that for simple linear models the input noise is transferred to the output. In this paper, we analyse regression methods with input noise assumption.

Input noise modelling is more difficult than modelling output or observational noise: whilst the output noise is directly observable, the input noise can be observed only through the input-output transformation that we *inferred*. In many cases there are analytically tractable solutions for the output noise model, while such solutions are extremely rare for the input noise, a notable exception is the restrictive linear model. Furthermore, even when the input noise can be integrated out analytically, the solution will be biased. As shown by [1], adding noise to the inputs can be used for regularization, but for non-zero curvatures, the estimated output is biased.

Consider for example the parabola in Figure 1, where the red circles on the horizontal axis are the noisy measurement locations in the neighbourhood of zero – Gaussian noise is assumed –, the blue stars are the regression values of the measurements – without output noise for clarity reasons – and the green point is the result of the Monte Carlo approximation of the integration. It is clear that after integration the prediction at zero will be always below the true value of the function, caused by the non-zero curvature of the function, e.g., for concave functions the averaging over the noisy measurements *pulls down* the prediction.

© Springer-Verlag Berlin Heidelberg 2013

The method proposed in this article improves regression with input noise by removing the bias induced by the input noise. We show that the bias is proportional to the variance of the noise and to the second order derivative (Hessian) of the objective function. The method can be used for any regression algorithm as long as its Hessian can be calculated.

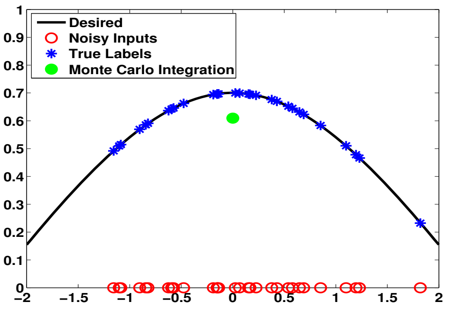


Fig. 1. The bias when estimating with input noise: the predicted value at zero will *always* be underestimated

1.1 Related Work

Most of the work with input noise has been done for specific models, e.g., Gaussian processes (GP), neural networks (NN), that we discuss later. An exception is [11] who showed that the regularization with input noise [1] is equivalent to adding the L_2 norm of the Hesse matrix of the objective function. Our method is based on this insight with reversed goals, i.e., we do not want more regularized models but better accuracy. In the GP framework a general idea is to use a second GP to model the input noise process. Posteriors have been obtained via Monte Carlo integration [6], variational learning [8], or EM-based methods [7]. Another approach is to use the Taylor expansion of the GP [5] and analytically compute the posterior (e.g., for squared exponential kernels) [2]. The integration of the input noise is intractable for NNs as well. To approximate the posterior, [12] integrated over the uncertain input with a maximum likelihood estimation, while [13] used Laplace approximation and Monte Carlo simulation to improve on the prediction.

The above approaches do not report improvement in the accuracy of the prediction, they focus instead on improved posterior variance estimates. Furthermore, most of them were applied for one dimensional problems, with only a few being used on real world data-sets with multidimensional inputs, e.g., [2,9,8]. We apply our method for both artificial problems and real-world data-sets.

2 Input Noise Correction

Let us be given a data-set $\boldsymbol{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, with inputs $\boldsymbol{x}_i \in \Re^d$ and labels $y_i \in \Re$. A general assumption is that the labels are corrupted with additive Gaussian noise, and we also assume that the inputs are also corrupted, i.e.,

$$y = \tilde{y} + \epsilon_y \qquad\qquad \boldsymbol{x} = \tilde{\boldsymbol{x}} + \boldsymbol{\epsilon}_x,$$

where y is the observed label, $\tilde{y}$ is the true label, and $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ is a noise process; $\boldsymbol{x}$ is the observed input, $\tilde{\boldsymbol{x}}$ is the true input, and $\boldsymbol{\epsilon}_x \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ is the additive Gaussian input noise. We assume that the inputs are uncorrelated, i.e., $\boldsymbol{\Sigma}$ is a diagonal covariance matrix with σ_i^2 on the diagonal and $\boldsymbol{\sigma}$ denotes the vector of individual variances. If we denote with $f(\cdot)$ the true data generating function, with $p(\boldsymbol{\epsilon}_x)$ the distribution of the input noise, and with $\hat{f}(\cdot)$ the function after integrating out the input noise, the relation between them is

$$\hat{f}(\tilde{\boldsymbol{x}}) = \int f(\tilde{\boldsymbol{x}} + \boldsymbol{\epsilon}_x)\, dp(\boldsymbol{\epsilon}_x), \tag{1}$$

The estimate $\hat{f}(\cdot)$ is biased even when the true generating function $f(\cdot)$ is known [12]. We propose to use the second order Taylor expansion of $f(\cdot)$ around the true input location $\tilde{\boldsymbol{x}}$, with the averaging as

$$\hat{f}(\tilde{\boldsymbol{x}}) = \int \left(f(\tilde{\boldsymbol{x}}) + \boldsymbol{\epsilon}_x^\top J_f(\tilde{\boldsymbol{x}}) + \frac{1}{2}\boldsymbol{\epsilon}_x^\top H_f(\tilde{\boldsymbol{x}})\boldsymbol{\epsilon}_x + \dots \right) dp(\boldsymbol{\epsilon}_x), \tag{2}$$

where $J_f(\boldsymbol{x})$ and $H_f(\boldsymbol{x})$ are the Jacobian and the Hessian of $f(\boldsymbol{x})$. The first term does not depend on the noise; the Jacobian term vanishes since $\boldsymbol{\epsilon}_x$ is has zero mean; and the third term can be written as $\boldsymbol{\epsilon}_x^\top H_f(\tilde{\boldsymbol{x}})\boldsymbol{\epsilon}_x = \mathrm{tr}(H_f(\tilde{\boldsymbol{x}})\, \boldsymbol{\epsilon}_x \boldsymbol{\epsilon}_x^\top)$, leading to the following simplified expression:

$$\hat{f}(\tilde{\boldsymbol{x}}) \simeq f(\tilde{\boldsymbol{x}}) + \frac{1}{2}\boldsymbol{\sigma}^\top H_f(\tilde{\boldsymbol{x}})\boldsymbol{\sigma} \simeq f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\sigma}^\top H_f(\boldsymbol{x})\boldsymbol{\sigma}, \tag{3}$$

The true input location $\tilde{\boldsymbol{x}}$ is unknown; we approximate it with the noisy location $\boldsymbol{x}$, i.e. $f(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}})$ and $H_f(\boldsymbol{x}) = H_f(\tilde{\boldsymbol{x}})$. A similar assumption has been made by [9] in the context of input noise GPs.

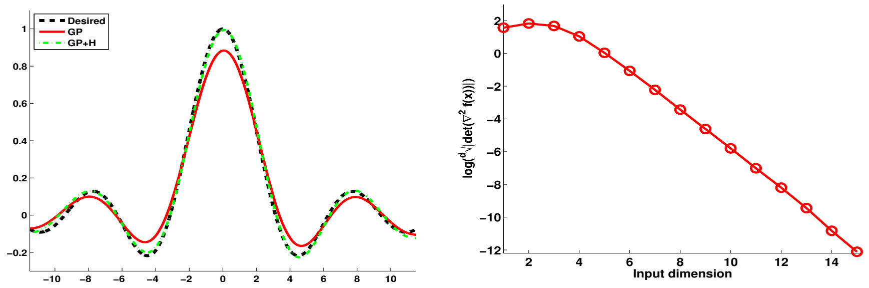
We consider two steps in approximating $\hat{f}(\tilde{\boldsymbol{x}})$: we first construct a function $g(\cdot)$ based on $\boldsymbol{D}$ *without* the input noise assumption. In this first step we do not make specific assumptions about the function, we assume that it can be arbitrary. From Equation (3) follows that $g(\cdot)$ and the *true* data generating function $f(\cdot)$ are related as follows:

$$g(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\sigma}^\top H_f(\boldsymbol{x})\boldsymbol{\sigma}. \tag{4}$$

The second step is obtaining $f(\cdot)$ when $g(\cdot)$ and $\boldsymbol{\sigma}$ are known, i.e., solving the partial differential equation from Equation (4). This equation does not have an analytical solution [3] since it requires an integration over $g(\boldsymbol{x})$ that is intractable in most cases. A possible solution is to use numerical methods but these methods time consuming and become unstable when noise is present. A more significant drawback is the lack of good initial conditions for the differential equation. Note that the initial conditions must contain both the values and the derivatives of the function $f(\cdot)$ [3]. We tried the following approximations for the initial conditions

$$f(\boldsymbol{x}) = g(\boldsymbol{x}),\;\; J_f(\boldsymbol{x}) = J_g(\boldsymbol{x}) \qquad \text{or} \qquad f(\boldsymbol{x}_j) = y_j,\;\; J_f(\boldsymbol{x}_j) = J_g(\boldsymbol{x}_j),$$

where $(\boldsymbol{x}_j, y_j) \in \boldsymbol{D}$ but our experiments show that pure results can be obtained based on these approximations. Next, we make further assumptions about $f(\cdot)$ to approximate the solution of the partial differential equation (4).



(a) Example for $f(x) = \mathrm{sinc}(x)$. (b) Curvature of the prediction.

Fig. 2. (a) Approximating with 800 training points and input noise with standard deviation $\sigma = 0.8$ the standard GP (red) under- and overestimates the true prediction (black) where the curvature is high. Our GP model corrected with the Hessian (green) results in more accurate prediction. (b) The curvature of the predicted function tends to zero exponentially when the dimension of the inputs increases.

2.1 Quadratic Approximation of the Partial Differential Equation

We approximate $f(\cdot)$ with a quadratic function at every input location $\boldsymbol{x}$, i.e. we assume that $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}_x \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{b} + c$ with its Hessian $H_f(\boldsymbol{x}) = 2\boldsymbol{A}_x$. This is again similar to the approximation based on the Taylor series expansion from Equation (2), and we assume that the expansion is at the current point of interest $\boldsymbol{x}$.

We substitute the local approximation into Equation (4), differentiate it two times and obtain

$$H_g(\boldsymbol{x}) = 2\boldsymbol{A}_x. \tag{5}$$

i.e. under the locally quadratic approximation of $f(\cdot)$, the Hessian of $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ *must be equal*. Therefore, we can replace $H_f(\boldsymbol{x})$ with $H_g(\boldsymbol{x})$ from Equation (4), and obtain the following expression for $f(\boldsymbol{x})$

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) - \frac{1}{2}\boldsymbol{\sigma}^\top H_g(\boldsymbol{x})\boldsymbol{\sigma} \tag{6}$$

On the right side of Equation (6) every term is known, thus $f(\boldsymbol{x})$ has an analytic form. The interpretation of Equation (6) is the following: when dealing with data corrupted with input noise, any regression model $g(\cdot)$ can be improved by subtracting the Hessian of the model $H_g(\cdot)$ multiplied by the noise variance. An other interpretation of the result in Equation (6) is that if we relax our assumptions about the model, then we might replace the second derivatives of the true function from Equation (4) with the approximating function $g(\cdot)$ and this approximation is pursued in the rest of the paper.

3 Experiments

We conducted experiments to see how well the proposed method performs under different conditions. We were interested how well it scales with the number of the training examples, with the variance of the input noise, and with the dimension of the input space. We did not compare the proposed method with the state-of-the-art input noise models since the authors do not report significant improvement on the accuracy of the prediction, rather they enhance the posterior variance that is not desired in our framework.

3.1 Illustration for the Sinc(x) Function

We applied the proposed method on a toy example, to give an insight about the induced improvement. For training inputs we generated 800 points on the $[-12, 12]$ interval corrupted with additive Gaussian noise with standard deviation $\sigma = 0.8$. For training labels we transformed these point with the sinc$(\cdot)$ function and added a Gaussian noise with standard deviation $\sigma_y = 0.1$ as output noise. We used standard GPs to obtain prediction and also used the Hessian corrected version (GP+H). Results are shown on Figure 2a. It can be seen that the standard GP under- or overestimates the true sinc$(\cdot)$ function where the curvature of the function is high. On the other hand, the Hessian corrected GP results in an almost perfect prediction. Note that when the variance of the input noise is not known, we can under- or over- correct the model using wrong values.

3.2 Synthetic Data

We generated synthetic data for different one dimensional functions, with results in Figures 3. We used GPs and NNs for the regression models to learn these functions and compared the accuracy with the Hessian corrected versions of the respective methods (referred as GP+H and NN+H). The hyper-parameters of the GP were obtained with evidence maximization while the parameters of the NN were obtained using back-propagation.

For every function we investigated (1) how the improvement of the input noise correction scales with increasing the standard deviation of the input noise, and (2) how the improvement of the input noise correction scales with increasing the number of the training examples with a fixed standard deviation. As a measure of performance we used the mean square error (MSE) of the learned function on the same interval where the training data was generated.

For the first type of experiments (first and third columns of Figures 3) we generated 200 points from the interval $[-4, 12]$ (or with the intersection where the respective function was defined) and added a Gaussian output noise with standard deviation $\sigma_y = 0.1$. The standard deviation of the input noise was between 0.1 and 1.2. For the second type of experiments (second and fourth columns of Figures 3) we generated points from the same interval as before, added a Gaussian input noise with standard deviation $\boldsymbol{\sigma} = 0.8$, and also added Gaussian output noise with standard deviation $\sigma_y = 0.1$. The number of the

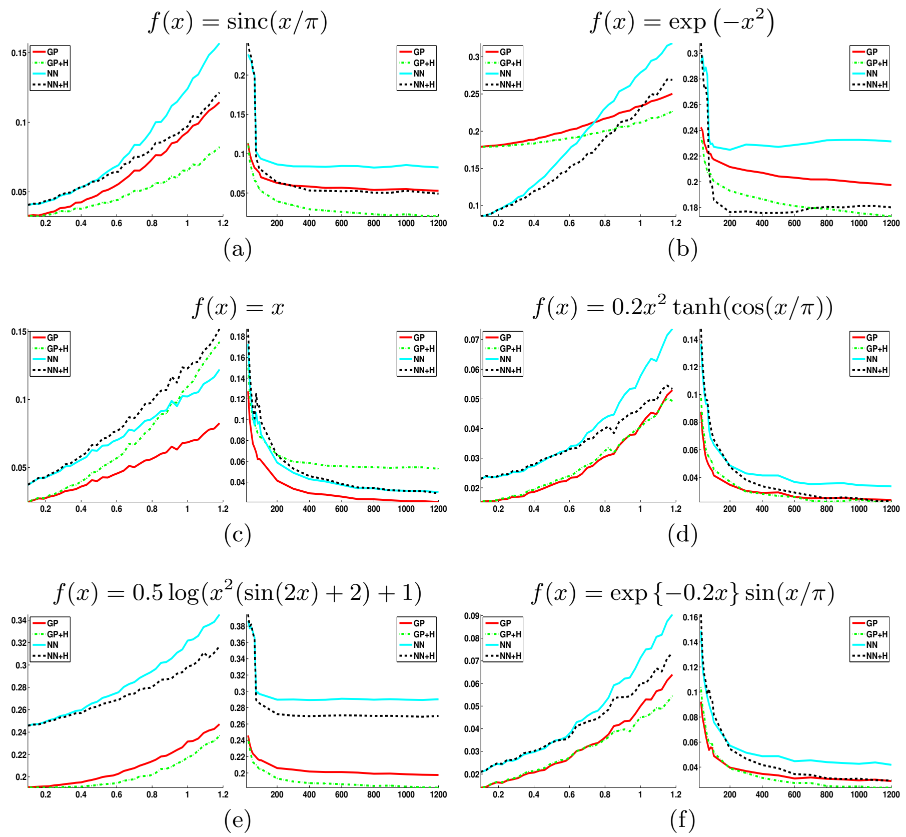


Fig. 3. Performance on artificial data: the left-side shows the evolution of the mean-square error when increasing gradually the input noise level. Plots on the right side show the errors when the size of the available data increases.

training points was between 20 and 1200 and all the shown results are averages over 200 runs.

Synthesizing plots of the algorithm, in Figure 3, show that the Hessian corrected version are almost always better than the standard GP or NN or at least it is very close. There is one exception: the linear function from Figure 3.(c), where the Hessian corrected version is significantly worse. The explanation is that the true Hessian of a linear model is zero. Thus, small inaccuracies in the standard regression model lead to non-zero Hessian, and therefore the prediction will deteriorate.

The general trend with increasing the noise level is that as the standard deviation of the input noise increases, the improvement induced by the Hessian corrected methods is more significant.

Another important conclusion is that as the number of the training examples grows, the MSE of the Hessian corrected estimation decreases (an exception is again the linear function), thus, it is consistent in the sense that it converges to the best model that the chosen function space contains.

Table 1. Results of experiments on real world data-sets (performance measured in MSE). Boston housing ($1000s); Concrete (mega-pascal); Barrett WAM (millimetres); CPU performance (benchmark points); Auto MPG (miles per gallon).

Data-set name	GP	GP+H	NN	NN+H	Noise (σ)	Dim.	Set size
Boston housing ($1000s)	2.2271	2.2271	3.4819	3.4819	0.1	13	506
Concrete (MPa)	4.1281	4.1280	6.1865	6.1864	1	8	1030
Barrett WAM 1 (mm)	2.9272	2.9242	2.8726	2.8488	0.01	4	1000
Barrett WAM 2 (mm)	13.707	13.731	12.439	9.3524	0.1	4	1000
CPU performance	17.762	17.763	16.846	16.846	5	7	209
Auto MPG (mpg)	4.0301	4.0322	2.2990	2.2988	3	7	301

3.3 Real World Data-Sets

When trying the method for real data, we were interested in how significant the Hessian corrected improvement is on higher-dimensional data where there is no control over the noise of the function to be predicted either. The data-sets were gathered from different domains with different features (input dimension, training set size) – Table 1 summarizes the data-sets. We used the (1) Boston housing data-set [4] that concerns housing values in suburbs of Boston; (2) Concrete data-set [14] that collected the compressive strength of the concrete; (3) Barrett whole arm manipulator (WAM) data-set [10] that was generated by us on a simulated Barrett WAM robot architecture while we learned the forward kinematics of the robot arm [10]; (4) CPU performance data-set [4] that deals with the CPU performance; (5) Auto MPG data-set [4] that collected fuel consumption of cars.

Results from Table 1 show the standard deviation of the input noise, the dimension of the data-set, and the size of the data-set as well. The values were obtained with 10-fold cross validation and are averages over 100 runs.

The improvement of the Hessian corrected methods is insignificant but it is generally not worse. We believe that the explanation is the same as it was for the linear functions in the previous section. In high dimensions we prefer rather linear models (e.g., as a result of regularization) to avoid over-fitting. In theory we prefer models with Hessians close to zero. Thus, the addition of the approximated Hessian does not improve the prediction.

To illustrate this effect of high dimensional data on the Hesse matrix, we experimented on a toy example. We approximated a d dimensional parabola $f(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2$ using a GP. We generated 100 points uniformly distributed on the interval $[-2, 2]^d$ and did not add any noise. Figure 2b shows that the curvature of the prediction function tends to zero exponentially when the dimension of the inputs increases. Note that this phenomena is independent of any of our assumptions, it is rather a property of high dimensional data modeling.

4 Discussion

When the inputs of a data-set are corrupted with noise, integrating out the noise process leads to biased estimates. We presented a method that corrects this bias.

The correction is proportional to the Hessian of the learned model and to the variance of the input noise. The method works for arbitrary regression models.

The proposed method has limitations: it does not improve prediction for high-dimensional problems, where the data are *implicitly* scarce. This is due to the fact that the estimated Hessian is considerably flattened, leading to no significant contribution to the overall output. To wisely choose when the Hessian correction can be used with success, these limitations have to be taken into account.

An interesting further research direction is to further analyse our algorithm specialised for the robotic Barrett WAM data. We believe that there are potential improvement capabilities since the size of the data-set is large, the dimensionality of the problem is reduced, and there is a real need for better approximation methods. One could – for example – start from the approximation to the second order PDE from Equation (6) and try to provide still approximating solutions that would probably be more precise than the simple replacement of the true function with its approximated based on noiseless inputs.

The authors acknowledge the support of the Romanian Ministry of Education, grant PN-II-RU-TE-2011-3-0278.

References

1. Bishop, C.M.: Training with noise is equivalent to Tikhonov regularization. Neural Computation 7(1), 108–116 (1995)
2. Dallaire, P., Besse, C., Chaib-draa, B.: An approximate inference with Gaussian process to latent functions from uncertain data. Neuroc. 74(11), 1945–1955 (2011)
3. Evans, L.: Partial Differential Equations. Graduate Studies in Mathematics. American Mathematical Society (2010)
4. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
5. Girard, A., Murray-Smith, R.: Learning a Gaussian process model with uncertain inputs. Tech. rep., University of Glasgow, Department of Computing Science (2003)
6. Goldberg, P.W., Williams, C.K.I., Bishop, C.M.: Regression with input-dependent noise: A Gaussian process treatment. In: NIPS (1997)
7. Kersting, K., Plagemann, C., Pfaff, P., Burgard, W.: Most likely heteroscedastic Gaussian process regression. In: ICML, pp. 393–400. ACM, New York (2007)
8. Lzaro-gredilla, M., Titsias, M.K.: Variational heteroscedastic Gaussian process regression. In: ICML, pp. 841–848. ACM (2011)
9. McHutchon, A., Rasmussen, C.E.: Gaussian process training with input noise. In: NIPS, pp. 1341–1349 (2011)
10. Nguyen-Tuong, D., Peters, J.: Incremental online sparsification for model learning in real-time robot control. Neurocomputing 74(11), 1859–1867 (2011)
11. Rifai, S., Glorot, X., Bengio, Y., Vincent, P.: Adding noise to the input of a model trained with a regularized objective. CoRR abs/1104.3250 (2011)
12. Tresp, V., Ahmad, S., Neuneier, R.: Training neural networks with deficient data. In: NIPS, pp. 128–135. Morgan Kaufman Publishers (1994)
13. Wright, W.: Neural network regression with input uncertainty. In: Neural Networks for Signal Processing VIII, pp. 284–293. IEEE (1998)
14. Yeh, I.C.: Modeling of strength of high performance concrete using artificial neural networks. Cement and Concrete Research 28(12), 1797–1808 (1998)

Model-Based Clustering of Temporal Data

Hani El Assaad[1], Allou Samé[1], Gérard Govaert[2], and Patrice Aknin[1]

[1] Université Paris-Est, IFSTTAR,
GRETTIA, F-77420 Champs-Sur-Marne, France
{hani.el-assaad,allou.same,patrice.aknin}@ifsttar.fr
[2] Université de technologie de Compiègne,
UMR CNRS 7253 Heudiasyc, F-60205 Compiègne, France
gerard.govaert@utc.fr

Abstract. This paper addresses the problem of temporal data clustering using a dynamic Gaussian mixture model whose means are considered as latent variables distributed according to random walks. Its final objective is to track the dynamic evolution of some critical railway components using data acquired through embedded sensors. The parameters of the proposed algorithm are estimated by maximum likelihood via the Expectation-Maximization algorithm. In contrast to other approaches as the maximum a posteriori estimation in which the covariance matrices of the random walks have to be fixed by the user, the results of the simulations show the ability of the proposed algorithm to correctly estimate these covariances while keeping a low clustering error rate.

Keywords: Clustering, dynamic latent variable model, mixture model, EM algorithm, Kalman filter, time series clustering, maximum likelihood, maximum a posteriori.

1 Introduction

Clustering, which consists of automatically identifying groups into data sets, remains a central problem in many applications including web data mining, marketing, bio-informatics, image segmentation. For mining independent numerical data, the Gaussian mixture model [5,10], used conjointly with the Expectation-Maximization [2], is now well known to provide an efficient clustering solution. However, some challenges still remains for issues of clustering non stationary data.

More particularly, this study was motivated by the characterization of the dynamic evolution of some critical railway components (point machines and doors systems) using condition measurements acquired through embedded sensors. Its final objective is to build a decision-aided support for their preventive maintenance. One of the difficulties in achieving this goal is that, during their dynamic evolution, these components may switch between different states due to various operating contexts (different hygrometric conditions, different levels of train inclinations).

© Springer-Verlag Berlin Heidelberg 2013

We propose to solve this problem by automatically extracting, from temporal data, clusters whose characteristics evolve over time. In this framework, the clusters can be interpreted as the states within the operating contexts. This dynamical clustering problem can be addressed by assuming that the data are distributed according to a Gaussian mixture model whose centers evolve linearly with time [3, 11]. However, a linear evolution of the clusters may turn out to be inefficient for complex non linear dynamics. A more appropriate modeling consists in assuming that the cluster centers are themselves distributed according to Gaussian random walks. Calabrese and Paninski [1] have proposed a Bayesian estimation of such mixture model for tracking time-varying spike shapes. In their formulation, the Gaussian random walks modeling the cluster centers were treated as prior probability distributions and therefore, their variances, as smoothing hyperparameters, had to be chosen adequately.

A new model formulation, together with its maximum likelihood parameter estimation through the EM algorithm, is introduced in this paper. Moreover, one of the key points of this approach remain to be its capability to estimate the random walks variances which control the temporal regularity of the clusters.

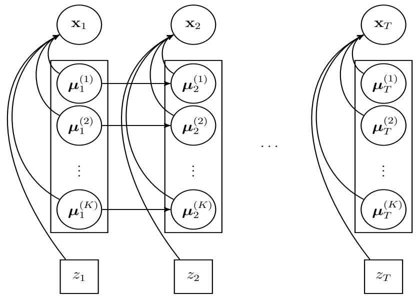
The rest of the paper is organized as follows. In Section 2, we formalize the dynamic model for clustering temporal data. We introduce the proposed maximum likelihood (ML) approach for parameters estimation in Section 3. Our approach is evaluated in Section 4 using simulated and real data. Conclusions and future works are given in Section 5.

2 Dynamic Model for Temporal Data Clustering

The observed data sequence to be classified will be denoted as $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ where $\mathbf{x}_t \in \mathbb{R}^d$, and the associated unobserved class will be denoted as $(z_1, \ldots, z_T)$, where $z_t \in \{1, \ldots, K\}$. According to the proposed model, the observed data is supposed to be sampled from K Gaussian distributions whose means evolve dynamically in the course of time. More formally, given the parameters vector $\boldsymbol{\Theta} = \{(\pi_k, \boldsymbol{\mu}_0^{(k)}, v_k^2, \sigma_k^2);\ k = 1, \ldots, K\}$ belonging to $\mathbb{R}^{K(d+3)-1}$, the data generation scheme is the following:

- the class z_t are independently drawn according to a multinomial distribution: $z_t \sim \mathcal{M}(1, \pi_1, \ldots, \pi_K)$, with $\pi_k = P(z_t = k)$ and $\sum_{k=1}^{K} \pi_k = 1$;
- the time dependent cluster centers $\boldsymbol{\mu}_t^{(k)} \in \mathbb{R}^d$ are generated according to Gaussian random walks with spherical covariance matrices $v_k^2 I$, where I is the identity matrix in $\mathbb{R}^d$: $\boldsymbol{\mu}_t^{(k)} \sim \mathcal{N}(\boldsymbol{\mu}_{t-1}^{(k)}, v_k^2 I)$,
- given the class z_t and the means $\boldsymbol{\mu}_t^{(z_t)}$, the observation $\mathbf{x}_t$ is generated according to a Gaussian distributions with mean $\boldsymbol{\mu}_t^{(z_t)}$ and covariance matrix $\sigma_{z_t}^2 I$: $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t^{(z_t)}, \sigma_{z_t}^2 I)$.

The graphical representation associated to this dynamic model is displayed in Figure 1. It should be noticed that this model is closely related to the switching



$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_T$
$\boldsymbol{\mu}_1^{(1)}$
$\boldsymbol{\mu}_1^{(2)}$
$\boldsymbol{\mu}_1^{(K)}$
$\boldsymbol{\mu}_2^{(1)}$
$\boldsymbol{\mu}_2^{(2)}$
$\boldsymbol{\mu}_2^{(K)}$
$\boldsymbol{\mu}_T^{(1)}$
$\boldsymbol{\mu}_T^{(2)}$
$\boldsymbol{\mu}_T^{(K)}$
z_1
z_2
z_T

Fig. 1. Graphical model representation of dynamic mixture model

state-space model introduced by Ghahramani and Hinton [9]. The main difference between the two models lies in the Markov property on the sequence $(z_1, \ldots, z_T)$, which is not assumed in our model.

3 Maximum Likelihood Estimation

In contrast to the maximum a posteriori estimation approach described in the previous section, which assumes a prior distribution over the clusters centers, we develop a new maximum likelihood approach which does not consider them as parameters but rather as random variables. As it will be detailed, the proposed approach is thus capable to estimate the variances v_k^2 of the random walks generating the cluster means.

We assume that the observed data is the sequence $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, the missing data consist of (a) the assignments $z = (z_1, \ldots, z_T)$ of the $\mathbf{x}_t$'s to the mixture components and (b) the cluster centers $\boldsymbol{\mu} = (\boldsymbol{\mu}_t^{(k)})$. Therefore, the parameter vector $\boldsymbol{\Theta}$ can be estimated by maximizing the log-likelihood function L_{ML} defined as follows:

$$\mathrm{L}_{\mathrm{ML}}(\boldsymbol{\Theta}) = \log p(\mathbf{x}; \theta) = \log \sum_{z} \int_{\boldsymbol{\mu}} p(\mathbf{x}, \boldsymbol{\mu}, z; \theta) \mathrm{d}\boldsymbol{\mu}. \quad (1)$$

The maximization of this log-likelihood can be performed by the EM algorithm [2,6]. The complete data log-likelihood, that will be denoted by $\mathrm{L}_{\mathrm{CML}}$, is written as:

$$\mathrm{L}_{\mathrm{CML}}(\boldsymbol{\Theta}) = \sum_{t=1}^{T} \sum_{k=1}^{K} z_{tk} \left(\log \pi_k \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t^{(k)}, \sigma_k^2 I) + \log \mathcal{N}(\boldsymbol{\mu}_t^{(k)}; \boldsymbol{\mu}_{t-1}^{(k)}, v_k^2 I) \right). \quad (2)$$

The two steps of the EM algorithm are introduced in the following sections.

3.1 E-Step

The expected complete log-likelihood can be written as:

$$Q(\boldsymbol{\Theta};\boldsymbol{\Theta}^{(c)}) = \mathrm{E}(\mathrm{L}_{\mathrm{CML}}(\boldsymbol{\Theta})|\mathbf{x};\boldsymbol{\Theta}^{(c)}) = Q_1((\pi_k)) + Q_2\big((\boldsymbol{\mu}_0^{(k)}, v_k^2)\big) + Q_3\big((\sigma_k^2)\big), \tag{3}$$

with

$$Q_1 = \sum_{t=1}^{T}\sum_{k=1}^{K} \tau_{tk}^{(q)} \log \pi_k, \tag{4}$$

$$Q_2 = -\sum_{t=1}^{T}\sum_{k=1}^{K} \frac{\tau_{tk}^{(q)}}{2}\left(p \log v_k^2 + \frac{1}{v_k^2}\mathrm{E}\left(\|\boldsymbol{\mu}_t^{(k)} - \boldsymbol{\mu}_{t-1}^{(k)}\|^2 \mid z_{tk}=1, \mathbf{x}; \boldsymbol{\Theta}^{(q)}\right)\right), \tag{5}$$

$$Q_3 = -\sum_{t=1}^{T}\sum_{k=1}^{K} \frac{\tau_{tk}^{(q)}}{2}\left(p \log \sigma_k^2 + \frac{1}{\sigma_k^2}\mathrm{E}\left(\|\mathbf{x}_t - \boldsymbol{\mu}_t^{(k)}\|^2 \mid z_{tk}=1, \mathbf{x}; \boldsymbol{\Theta}^{(q)}\right)\right). \tag{6}$$

where $\tau_{tk}^{(q)} = P(z_{tk}=1|\mathbf{x}_1,\ldots,\mathbf{x}_T;\boldsymbol{\Theta}^{(q)})$ is the posterior probability that $\mathbf{x}_t$ originates from cluster k, given the parameter vector $\boldsymbol{\Theta}^{(q)}$. Unfortunately, the posterior assignment probability $\tau_{tk}^{(q)}$ is not straightforward to compute. Indeed, it requires successive integrals over mixture of normal distributions that become intractable. We suggest approximating this probability by

$$\tau_{tk}^{(q)} \approx \frac{\pi_k^{(q)}\mathcal{N}(\mathbf{x}_t;\mathbf{c}_t^{(k)},\sigma_k^{2^{(q)}} I)}{\sum_{\ell=1}^{K}\pi_\ell^{(q)}\mathcal{N}(\mathbf{x}_t;\mathbf{c}_t^{(\ell)},\sigma_\ell^{2^{(q)}} I)}, \tag{7}$$

approximation that was found to work well on different synthetic data. To evaluate the quantities Q_2 and Q_3, let us define the expectations and the variances $\mathbf{c}_{tk}^{(q)} = \mathrm{E}(\boldsymbol{\mu}_t^{(k)}|\mathbf{x}_1,\ldots,\mathbf{x}_T, z_{tk}=1;\boldsymbol{\Theta}^{(q)})$, $P_{tk}^{(q)} = \mathrm{var}(\boldsymbol{\mu}_t^{(k)}|\mathbf{x}_1,\ldots,\mathbf{x}_T, z_{tk}=1;\boldsymbol{\Theta}^{(q)})$ and $P_{t,t-1,k}^{(q)} = \mathbf{cov}(\boldsymbol{\mu}_t^{(k)},\boldsymbol{\mu}_{t-1}^{(k)}|\mathbf{x}_1,\ldots,\mathbf{x}_T, z_{tk}=1;\boldsymbol{\Theta}^{(q)})$, which can be computed using the following weighted Kalman filtering and smoothing recursions [4,8]:

- ***Forward (filtering)***: starting from $\mathbf{m}_{0k} = (\boldsymbol{\mu}_0^{(k)})^{(q)}$ and $\mathbf{M}_{0k} = \mathbf{0}$, compute, for $t=1,\ldots,T$,

$$K_{tk} = (\mathbf{M}_{t-1,k} + v_k^{2^{(q)}} I)(\mathbf{M}_{t-1,k} + v_k^{2^{(q)}} I + (\sigma_k^{2^{(q)}}/\tau_{tk}^{(q)})\, I)^{-1},$$

$$\mathbf{m}_{tk} = \mathbf{m}_{t-1,k} + K_{tk}(\mathbf{x}_t - \mathbf{m}_{t-1,k}) \quad \text{and} \quad \mathbf{M}_{tk} = (I - K_{tk})(\mathbf{M}_{t-1,k} + v_k^{2^{(q)}} I).$$

- ***Backward (smoothing)***: starting from $P_{T,T-1,k}^{(q)} = (I - K_{Tk})\mathbf{M}_{T-1,k}$ and $\mathbf{c}_{Tk}^{(q)} = \mathbf{m}_{Tk}$, compute, for $t = T-1,\ldots,1$,

$$J_{tk} = \mathbf{M}_{tk}(\mathbf{M}_{tk} + v_k^{2^{(q)}} I)^{-1},$$
$$\mathbf{c}_{tk}^{(q)} = \mathbf{m}_{tk} + J_{tk}(\mathbf{c}_{t+1,k}^{(q)} - \mathbf{m}_{tk}),$$
$$P_{tk}^{(q)} = \mathbf{M}_{tk} + J_{tk}\left(P_{t+1,k}^{(q)} - (\mathbf{M}_{tk} + v_k^{2^{(q)}} I)\right) J_{tk}',$$

  and compute for $t = T-1,\ldots,2$,

$$P_{t,t-1,k}^{(q)} = \mathbf{M}_{tk}\, J_{t-1,k}' + J_{tk}(P_{t+1,t,k}^{(q+1)} - \mathbf{M}_{tk})J_{t-1,k}'.$$

Finally, we get:

$$Q_2 = -\sum_{t=1}^{T}\sum_{k=1}^{K}\frac{\tau_{tk}^{(q)}}{2}\left(p\log\sigma_k^2 + \frac{1}{\sigma_k^2}\left(\|\mathbf{x}_t - \mathbf{c}_t^{(k)}\|^2 + \mathbf{tr}(P_t^{(k)})\right)\right), \tag{8}$$

$$\begin{aligned}Q_3 = &-\sum_{k=1}^{K}\frac{\tau_{1k}^{(q)}}{2}\left(p\log v_k^2 + \frac{1}{v_k^2}\left(\|\mathbf{c}_1^{(k)} - \boldsymbol{\mu}_0^{(k)}\|^2 + \mathbf{tr}(P_1^{(k)})\right)\right)\\ &-\sum_{t=2}^{T}\sum_{k=1}^{K}\frac{\tau_{tk}^{(q)}}{2}\left(p\log v_k^2 + \frac{1}{v_k^2}\|\mathbf{c}_t^{(k)} - \mathbf{c}_{t-1}^{(k)}\|^2\right)\\ &-\sum_{t=2}^{T}\sum_{k=1}^{K}\frac{\tau_{tk}^{(q)}}{2v_k^2}\mathbf{tr}\left(P_t^{(k)} - P_{t,t-1}^{(k)} - (P_{t,t-1}^{(k)})' + P_{t-1}^{(k)}\right).\end{aligned} \tag{9}$$

3.2 M-Step

In the M-step, we have to separately maximize the quantities Q_1, Q_2 and Q_3. The updated parameters are then given by:

$$(\boldsymbol{\mu}_0^{(k)})^{(q+1)} = \mathbf{c}_{1k}^{(q)}, \qquad \pi_k^{(q+1)} = \frac{\sum_{t=1}^{T}\tau_{tk}^{(q)}}{T} \tag{10}$$

$$\sigma_k^{2(q+1)} = \frac{\sum_{t=1}^{T}\tau_{tk}^{(q)}\left(\|\mathbf{x}_t - \mathbf{c}_{tk}^{(q)}\|^2 + \mathbf{tr}(P_{tk}^{(q)})\right)}{p\sum_{t=1}^{T}\tau_{tk}^{(q)}}, \tag{11}$$

$$\begin{aligned}v_k^{2(q+1)} = &\frac{\sum_{t=2}^{T}\tau_{tk}^{(q)}\|\mathbf{c}_{tk}^{(q)} - \mathbf{c}_{t-1,k}^{(q)}\|^2 + \tau_{1k}^{(q)}\mathbf{tr}(P_{1k}^{(q)})}{p\sum_{t=2}^{T}\tau_{tk}^{(q)}} +\\ &\frac{\mathbf{tr}\left(\sum_{t=2}^{T}\tau_{tk}^{(q)}(P_{tk}^{(q)} - P_{t,t-1,k}^{(q)} - (P_{t,t-1,k}^{(q)})' + P_{t-1,k}^{(q)})\right)}{p\sum_{t=2}^{T}\tau_{tk}^{(q)}}.\end{aligned} \tag{12}$$

4 Experimental Study

In this section, we began by testing the performances of EM-ML on several example synthetic data sets. In this case, the knowledge of the ground truth allowed us to compare our algorithm to others algorithms. We generated 50 data sequences of length $T = 500$ according to the scheme described in section 2 with $K = 2$ clusters. The parameters used are as follows: $\pi_1 = \pi_2 = \frac{1}{2}$, $\sigma_1^2 = \sigma_2^2 = \frac{1}{4}$, $v_1^2 = v_2^2 = \frac{1}{16}$, $\boldsymbol{\mu}_0^{(1)} = 0$ and $\boldsymbol{\mu}_0^{(2)} = 4$. Initially, the clusters are well-separated but their trajectories may cross after some time. Figure 2 shows an example of data simulated according to this model.

The performance of the compared algorithms is measured using the mean square error $\mathbf{C} = \frac{1}{KT}\sum_{k=1}^{K}\sum_{t=1}^{T}(\boldsymbol{\mu}_t^{(k)} - \hat{\boldsymbol{\mu}}_t^{(k)})^2$, where $\boldsymbol{\mu}_t^{(k)}$, $\hat{\boldsymbol{\mu}}_t^{(k)}$ are true and estimated cluster centers, which represents the error between the estimated means and those simulated. Notice that, given the estimated parameters, the cluster centers trajectories are computed via the filtering and smoothing recursions.

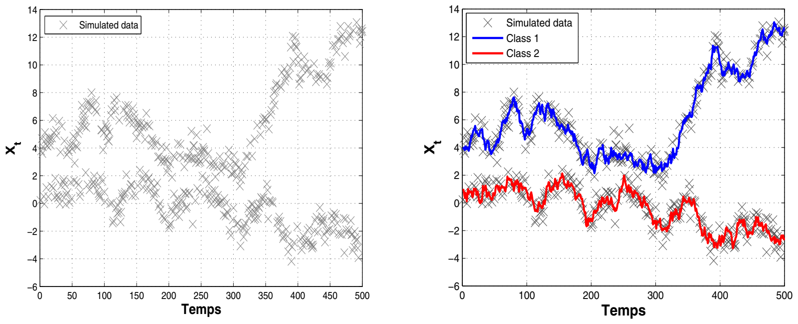


Fig. 2. Example of simulated data set

4.1 Algorithms in Competition

The following algorithms configurations are compared:

- EM-ML: the proposed maximum likelihood approach via the EM algorithm where the variances v_k^2 are automatically estimated,
- EM-MAP(v_k^2): the EM algorithm of Calabrese and Paninski [1] based on maximum a posteriori approach. This approach requires to set the value of the variance v_k^2 that we selected in $S = \{0.25; 0.15; 0.1; 0.05; 0.01\}$,
- EM-ML(v_k^2): the proposed algorithm run with fixed values of the variances $v_k^2 \in S$. It should be noted here that this version of the EM-ML algorithm, operating with fixed values of v_k^2, is only used to observe the performances of our approach when it is placed in the same conditions as the EM-MAP algorithm,
- theoretical: the EM-ML algorithm run using the true initial parameters.

For each simulated data set, these four algorithms were applied with the correct number of clusters $K = 2$.

4.2 Results

Figure 3 displays the criterion **C** (averaged over the 50 data sets) for each of the four algorithms configurations. The performances of EM-ML(v_k^2) and EM-MAP(v_k^2) evolve as a function of v_k^2, while those of EM-ML which does not depend on v_k^2 are represented by a horizontal line. Not surprisingly, the best results are obtained with the "theoretical" approach, which can thus be considered as the reference result. We observe that EM-ML(v_k^2) gives better results than EM-MAP(v_k^2). Even if EM-ML is outperformed by both EM-ML(v_k^2) and EM-MAP(v_k^2) for v_k^2 near the true value 0.06, its results are globally correct considering the fact that it automatically estimates the variances v_k^2.

Finally, we tested our method in terms of tracking the evolution of railway components, using a database of real signals issued from the switch operations.

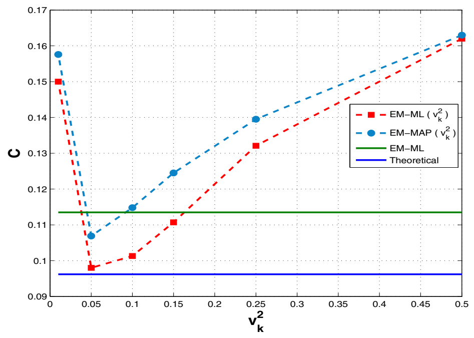


Fig. 3. Criterion **C** in relation to v_k^2 for the four algorithm configurations

More particularly, the goal will be to discover $K = 2$ operating context within the data (e.g., different hygrometric and temperature conditions). We consider a sequence of $n = 850$ real switch operation signals displayed on figure 4(a). The specificity of the curves to be analyzed in this context is that they are subject to five changes in regime (starting phase, unlocking phase, translation, point locking and friction phase) as a result of five successive mechanical movements of the physical components associated with the switch mechanism. Due to this specificity, the first step of the process consists in automatically modelling and segmenting each curve using a specific regression model [12]. In other words, this step consists in representing each power consumption curve by a mixture of five polynomial regression models (see [12] for more details). From this representation, each curve is summarized by the average value of its polynomial curves over the unlocking, translation and locking segments. The proposed algorithm has been run on the resulting time series described by figure 4(b) by setting $K = 2$. The clustering results are displayed figure on 4(c).

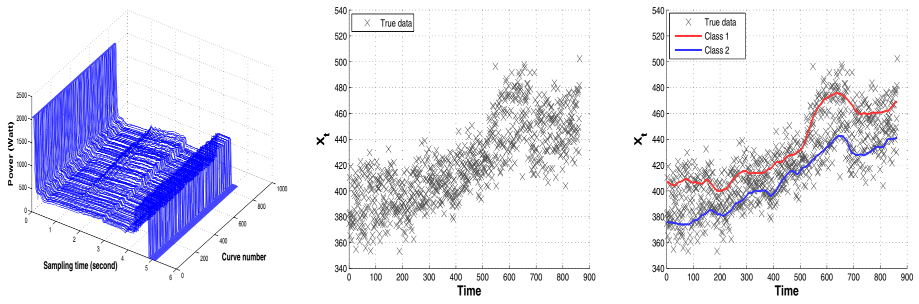


Fig. 4. (left) Examples of signals acquired during successive switch operations; (middle) Real data without the trajectories; (right) Clustering results obtained with EM-ML for the real data set.

5 Conclusion and Future Work

A new approach dedicated to temporal data clustering is proposed in this article. The dynamical model associated to this approach assumes that the cluster centers are latent random variables which evolve in the course of time according to random walks. Its parameters are learned by a maximum likelihood approach through the EM algorithm. In contrast to other approaches as the one based on a MAP estimation of parameters, the strength of our approach rely on its capability to estimate the covariance matrix of the random walks.

The experimental study conducted on synthetic and real switch operation signals data has shown encouraging results in terms of tracking the evolution of railway components under different operating contexts. The proposed EM-ML algorithm is capable to estimate the variances of the random walks while keeping a low error rate regarding the difference between the estimated cluster centers and those simulated. The main prospect of this work will be to apply the proposed algorithm to real data from railway transportation systems whose dynamic includes switching between various states related to operating contexts. We also plan to test Gibbs sampling approach for the calculation of the posterior assignment probability.

References

1. Calabrese, A., Paninski, L.: Kalman filter mixture model for spike sorting of non-stationary data. Journal of Neurosciences Methods 196(1), 159–169 (2011)
2. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B 39, 1–38 (1977)
3. DeSarbo, W.S., Cron, W.L.: A maximum likelihood methodology for clusterwise linear regression. Journal of Classification 5, 249–282 (1988)
4. Durbin, J., Koopman, S.J.: Time series analysis by state space methods. Oxford University Press (2001)
5. McLachlan, G.J., Peel, D.: Finite Mixture Models. Wiley, New York (2000)
6. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions. Wiley, New York (1997)
7. Jazwinski, A.H.: Stochastic Processes and Filtering Theory, pp. 201–217. Academic Press, New York (1970)
8. Shumway, R.H., Stoffer, D.S.: Time series analysis and its applications. Springer (2011)
9. Ghahramani, Z., Hinton, G.E.: Variational learning for switching state-space models. Neural Computation 12, 963–996 (1998)
10. Titterington, D.M., Smith, A.F., Makov, U.E.: Statistical Analysis of Finite Mixture Distributions. Wiley, New York (1985)
11. Wedel, M., DeSarbo, W.S.: A maximum likelihood approach for generalized linear models. Journal of Classification 12, 1–35 (1995)
12. Chamroukhi, F., Samé, A., Govaert, G., Aknin, P.: Time series modeling by a regression approach based on a latent process. Neural Networks 22(5-6), 593–602 (2009)

Fast Approximation Method for Gaussian Process Regression Using Hash Function for Non-uniformly Distributed Data

Yuya Okadome[1], Yutaka Nakamura[1,3], Yumi Shikauchi[2,3], Shin Ishii[2,3], and Hiroshi Ishiguro[1]

[1] Graduate School of Engineering Science, Osaka University
[2] Graduate School of Informatics, Kyoto University
[3] ATR Cognitive Mechanisms Laboratories
{okadome.yuya,nakamura}@irl.sys.es.osaka-u.ac.jp,
yumi-s@sys.i.kyoto-u.ac.jp,
ishii@i.kyoto-u.ac.jp,
ishiguro@sys.es.osaka-u.ac.jp

Abstract. Gaussian process regression (GPR) has the ability to deal with non-linear regression readily, although the calculation cost increases with the sample size. In this paper, we propose a fast approximation method for GPR using both locality-sensitive hashing and product of experts models. To investigate the performance of our method, we apply it to regression problems, i.e., artificial data and actual hand motion data. Results indicate that our method can perform accurate calculation and fast approximation of GPR even if the dataset is non-uniformly distributed.

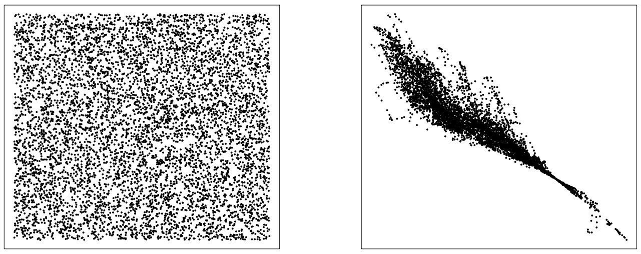
Keywords: Gaussian process regression, locality-sensitive hashing, product of experts model.

1 Introduction

Since Gaussian process regression (GPR)[11][4] can readily deal with non-linear regression and construct a prediction model flexibly, it has been studied in many fields[7][8][15]. Applications of a non-parametric model had been difficult due to high computational cost but they have become popular thanks to recent high performance computers and computationally efficient algorithms[9][13][2]. Although computational performance is improved, essential calculation cost increases dramatically for problems of processing datasets with large sample size.

Indyk et al. proposed a computationally efficient method for one of the non-parametric models, i.e. k-nearest neighbors, called Locality-Sensitive Hashing (LSH)[6][3], which is a technique for dividing the dataset into small subsets while preserving the locality of the data using a hash-key composed of outputs of binary threshold functions. The partition of the dataset can be done by calculating a hash-key value, and only the samples possessing hash-keys identical to the query

© Springer-Verlag Berlin Heidelberg 2013



(a) Dataset (uniform) (b) Dataset (non-uniform)

Fig. 1. examples of dataset

input are used to find nearest neighbors. As a result, the calculation cost is significantly reduced.

In this research, we derive a fast approximation method for GPR based on the above idea. However, since the accuracy of approximation gets worse near the borders between subspaces when the dataset is simply divided, we propose a method based on a product-of-experts model[5] to combine multiple decomposed GPR with different hash-keys. By compensating for the worse estimation of decomposed GPR, the accuracy is expected to improve.

Since the calculation cost is proportional to $O(n^2)$, it is better to divide the dataset into subsets with equal number of samples. In a naive application of LSH, it is assumed that the samples are distributed uniformly in the input space as shown in Fig. 1(a); however that is usually not the case in the distribution of an actual dataset. Furthermore, a set of simple threshold functions does not necessarily divide the dataset equally, especially when the distribution of samples is not uniform, as shown in Fig 1(b). Therefore, we try to solve this problem by using a tree-structured hash-key. We apply our method to the regression task of function approximation (artificial data) and motion data of the human hand (real data) and investigate its performance.

2 Gaussian Process Regression Approximated by the Divided Dataset

It is assumed that a dataset $D = (\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{X} = \{\boldsymbol{x}_n | n = 1, \ldots, N\}, \boldsymbol{y} = \{y_n | n = 1, \ldots, N\}$ is given. The purpose of a regression problem is to estimate the prediction function $y = f(\boldsymbol{x})$ from this dataset. GPR assumes that the prior distribution over the function is a Gaussian distribution. Then, outputs $\boldsymbol{f} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N))$ for input set $\boldsymbol{X}$ are defined as

$$P(\boldsymbol{f}|\boldsymbol{C},\boldsymbol{X}) = \frac{1}{Z}\exp\left[-\frac{1}{2}\boldsymbol{f}^\top\boldsymbol{C}^{-1}\boldsymbol{f}\right], \tag{1}$$

where $\boldsymbol{C}$ denotes the covariance matrix whose elements are $C_{ij} = K(\boldsymbol{x}_i,\boldsymbol{x}_j) + \delta_{ij}N(\boldsymbol{x}_i)$. $K(\cdot,\cdot)$ and $N(\cdot)$ denote a kernel function and a noise function, respectively. When a query input $\boldsymbol{x}_*$ is given, the output: $y_* = f(\boldsymbol{x}_*)$ is calculated as

$$P(y_*|\boldsymbol{x}_*,D) = \frac{1}{Z}\exp\left(-\frac{1}{2}\frac{(y_* - E[y_*])^2}{V[y_*]}\right). \tag{2}$$

As a result, the expectation and variance of the output become $E[y_*] = \boldsymbol{k}^\top\boldsymbol{C}^{-1}\boldsymbol{y}$ and $V[y_*] = K(\boldsymbol{x}_*,\boldsymbol{x}_*) - \boldsymbol{k}^\top\boldsymbol{C}^{-1}\boldsymbol{k}$. $\boldsymbol{k}$ is defined as $\boldsymbol{k} = (K(\boldsymbol{x}_1,\boldsymbol{x}_*),\ldots,K(\boldsymbol{x}_N,\boldsymbol{x}_*))$.

In order to calculate Eq. (2), it is necessary to calculate the matrix inverse ($\propto O(n^3)$) for the learning and the matrix multiplication ($\propto O(n^2)$) for each prediction. To reduce the calculation cost, we employ a hash-key to divide the dataset into multiple subsets. The hash-key corresponds to the subspace to which each subset belongs, and the calculation cost of this hash-value does not directly depend on the sample size.

LSH is usually used for a nearest neighbor search problem. Since it may happen that some neighbor points do not belong to the selected subset, L different hash-keys are used to reduce the number of such cases. In analogy with this idea, we use multiple decomposed GPRs possessing different hash-keys, and the total output is calculated using all the outputs of the decomposed GPR based on the product-of-experts model (PoEs)[5].

2.1 Approximated Calculation of GPR

To divide the dataset, a B-bits binary hash-key composed of B binary hash functions is used. All elements in each subset have an identical hash-key value. Since the calculation of the hash-key value does not depend on the sample size, the calculation cost for the partition is $O(1)$.

In this research, we assume that the value of the kernel function $K(\boldsymbol{x}_i,\boldsymbol{x}_*)$ between samples with different hash-key values ($g(\boldsymbol{x}_i) \neq g(\boldsymbol{x}_*)$) is small (e.g. RBF kernel) enough to be approximated by 0. The value of the kernel functions, $\boldsymbol{k}$, for the query input $\boldsymbol{x}_*$ is approximated as

$$\boldsymbol{k} \approx (0,0,\cdots,\underbrace{K(\boldsymbol{x}_i,\boldsymbol{x}_*),K(\boldsymbol{x}_{i+1},\boldsymbol{x}_*),\cdots}_{:=\boldsymbol{k}'},0,0,\cdots), \tag{3}$$

where only samples with hash-key value identical to the query input have non-zero values.

Similarly, the covariance matrix $\boldsymbol{C}$ can be approximated as a block diagonal matrix:

$$\hat{\boldsymbol{C}} = \begin{pmatrix} \hat{\boldsymbol{C}}_1 & & O \\ & \ddots & \\ O & & \hat{\boldsymbol{C}}_{2^B} \end{pmatrix}, \tag{4}$$

By using these equations, the expectation and variance of the prediction distribution of GPR are calculated as $\hat{E}[y_*] = \boldsymbol{k}'^\top \hat{\boldsymbol{C}}^{-1}_{g(\boldsymbol{x}_*)} \boldsymbol{y}_{g(\boldsymbol{x}_*)}$ and $\hat{V}[y_*] = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}'^\top \hat{\boldsymbol{C}}^{-1}_{g(\boldsymbol{x}_*)} \boldsymbol{k}'$. $\hat{\boldsymbol{C}}_{g(\boldsymbol{x}_*)}$ and $\boldsymbol{y}_{g(\boldsymbol{x}_*)}$ are the covariance matrix and the output data vector composed of samples whose hash-key value is $g(\boldsymbol{x}_*)$. Note that all $\hat{\boldsymbol{C}}_g$ and $\boldsymbol{y}_g$ can be calculated before the query input is given.

Tree-structured hash-key When the dataset is divided into equal-size subsets, it is expected that the calculation cost significantly reduces. However, if there are some large subsets, the calculation cost is dominated by such subsets and the computational efficiency can not be significantly improved. The naive LSH does not necessarily divide the dataset into equal-size subsets, especially when the distribution of the original data is non-uniform. We propose a tree structured hash-key to make the sample size of each subset equal. In this research, we use the random-projection tree[12] to generate a tree-structure hash-key.

2.2 Integration of Multiple Decomposed GPR by PoE

The precision in the border areas might decrease due to the shrinking of the efficient sample size. Employing multiple GPRs with different boundaries, and integrating their outputs to compensate each other seems to be a good way to cope with this problem. In this research, the calculation of the total GPR is performed on L independently decomposed GPRs with different hash-keys, i.e., different boundaries in the input space.

The prediction distribution of GPR (Eq. (1)) can be decomposed as

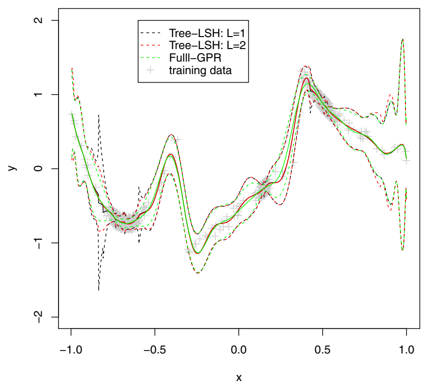
$$\begin{aligned} p(\boldsymbol{f}|\boldsymbol{C}, \boldsymbol{X}) &= \frac{1}{Z} \exp\left[-\frac{1}{2}\boldsymbol{f}^\top L(L\boldsymbol{C})^{-1}\boldsymbol{f}\right] = \frac{1}{Z}\prod_{l=1}^{L} \exp\left[-\frac{1}{2}\boldsymbol{f}^\top (L\boldsymbol{C})^{-1}\boldsymbol{f}\right] \\ &\approx \frac{1}{Z}\prod_{l=1}^{L} \exp\left[-\frac{1}{2}\boldsymbol{f}^\top (L\hat{\boldsymbol{C}}_l)^{-1}\boldsymbol{f}\right], \end{aligned} \tag{5}$$

where $\hat{\boldsymbol{C}}_l$ denotes the covariance matrix obtained by the l-th hash-key. Eq. (5) becomes equivalent to the original probabilistic model when approximation by decomposition does not produce any degradation.

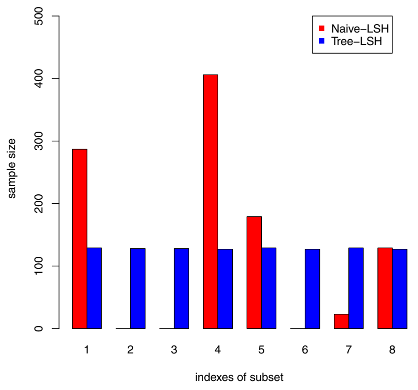
The prediction distribution of $\boldsymbol{y}_*$ for query $\boldsymbol{x}_*$ is calculated as

$$\begin{aligned} p(y_*|\boldsymbol{x}_*, D) &\approx \frac{1}{Z}\prod_{l=1}^{L} p(y_*|\boldsymbol{x}_*, D_{g_l(\boldsymbol{x}_*)}) \\ &= \frac{1}{Z}\exp\left(-\frac{1}{2}\sum_{l=1}^{L} \frac{(y_* - \hat{E}[y_*]_l)^2}{L\hat{V}[y_*]_l}\right), \end{aligned} \tag{6}$$

where $D_{g_l(\boldsymbol{x}_*)}$, $\hat{E}[y_*]_l$ and $\hat{V}[y_*]_l$ denote the selected subset, the predicted mean and the predicted variance using the l-th hash-key, respectively. The total mean



(a) Regression result



(b) The sample size of each subset

Fig. 2. Regression experiment using artificial data

$\overline{E}[y_*]$ and variance $\overline{V}[y_*]$ can be calculated by $\frac{\sum_{l=1}^{L} \hat{E}[y_*]_l \frac{1}{L\hat{V}[y_*]_l}}{\sum_{i=1}^{L} \frac{1}{L\hat{V}[y_*]_i}}$ and $\frac{1}{\sum_{l=1}^{L} \frac{1}{L\hat{V}[y_*]_l}}$, respectively.

The calculation cost of our method is $O(L2^B(N/2^B)^3)$ for inverse matrix and $O(L(N/2^B)^2)$ for prediction when the dataset is divided into equal size subsets. Also, the required amount of memory is $O(2^B(N/2^B)^2)$ in our method, as compared to $O(N^2)$ in a naive GP.

3 Performance Comparison

To investigate the performance of our method, we conducted a regression task using both artificial and real data.

3.1 Artificial Data

The plus marks (+) in Fig. 2(a) show the training dat; the sample size is 1024. The training data is distributed according to a Gaussian mixture and is concentrated to some small areas. The predictions are done by three types of GPR: speeding up by tree-structured hash-key (Tree-LSH-GPR), speeding up by naive LSH (Naive-LSH-GPR) and normal GPR (Full-GPR). The bit length of hash-keys is $B = 3$ and the number of keys is $L = 1, 2$ when an LSH is used. The accuracy of regression does not change when the number of hash-key becomes $L > 2$.

The green line, black line and red line in Fig. 2(a) show the approximation result of Full-GPR and Tree-LSH-GPR with $L = 1$ and $L = 2$, respectively.

Table 1. Computational time of the function approximation

	Prediction [ms]	Inverse [s]
Full-GPR	45.3 ± 1.2	12.4
Naive-LSH-GPR: $L = 1$	3.88 ± 2.24	0.160
Naive-LSH-GPR: $L = 2$	11.5 ± 3.7	0.462
Tree-LSH-GPR: $L = 1$	0.87 ± 0.004	0.0395
Tree-LSH-GPR: $L = 2$	1.76 ± 0.05	0.0727

Table 2. nMSE and computational time of the regression

	gp-map-1 [16]	Full-GPR	Tree-LSH-GPR
nMSE	0.036*	0.033	0.034
Time [s]	N/A*	2.09	0.100

The sample size of gp-map-1 is 1024 and results (*) are adopted from [16]. In our method, the sample size of the training data is 7168 and the bit length and the number of hash-key were 3 and 3.

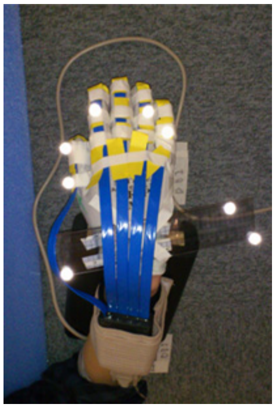
Fig. 2(b) shows the sample size of each subset for Tree-LSH-GPR and Naive-LSH-GPR. The red bar and blue bar show the divided result of Naive-LSH-GPR and Tree-LSH-GPR. Tree-LSH-GPR can divide the dataset into equal-size subsets. On the other hand, the sample sizes obtained by Naive-LSH-GPR are quite different.

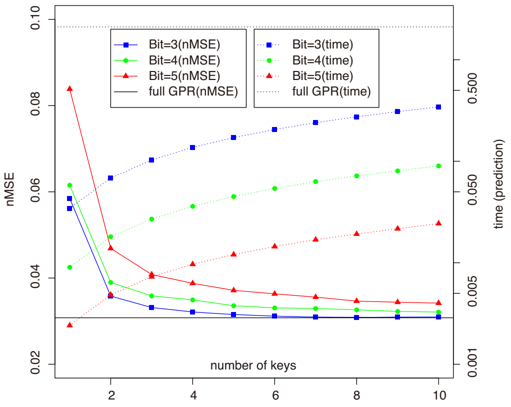
Table 1 shows the computational time for prediction and inverse matrix. Both Naive-LSH-GPR and Tree-LSH-GPR can calculate faster than the Full-GPR. Although the computational time by Naive-LSH-GPR with $L = 1$ was reduced to 1/10 for prediction and 1/80 for inverse matrix compared to Full-GPR, it has large variance. Tree-LSH-GPR further reduces the computational cost, and has small variance. Since the accuracy of the prediction did not differ much from that of Full-GPR, it can be said that Tree-LSH-GPR is more efficient. Note that by tuning the bit length and the number of hash-keys, we can balance the calculation cost and the accuracy.

Multi-dimensional Function Approximation. Table 2 shows the nMSE and computational time for a benchmark test used in [16], Pumadyn-8nm. The input dimensionality is 8 and the nonlinearity of the dataset is high. Due to the complexity of the dataset, gp-map-1 is one of the highest performance models among several supervised learning methods. Compared to this method with our method, we can use a large amount of dataset for the training and the performance can be improved, while the calculation time is reduced.

3.2 Regression Task with Real Data

We applied our method to a regression problem using actual experimental data, i.e. hand motion data. In the experiment, 8 markers attached on the hand


(a) data glove



(b) nMSE and prediction time

Fig. 3. Regression experiments using hand motion data

(Fig. 3(a)) and finger joint are recorded at the same time, using an optical motion capture system (Radish, Library) and a data glove (ShapeHand, Measurand). In an actual situation, it is not guaranteed that all sensors are available, so the purpose of the regression problem in this section is to predict the output value of one sensor from the output value of the other sensors. Each dataset sample consists of the input variable (positions of 5 markers attached on fingers, whose coordinate system is defined by the remaining 3 markers) and the output variable (the joint angle of the wrist).

The sample size of the dataset is 8192. We evaluate the performance by 8-fold cross-validation (7168 training data and 1024 test data). We compare the normal Gaussian process regression (Full-GPR) and the proposed method (Tree-LSH-GPR-B-L, $B = 3, 4, 5$,$L = 1, 2, .., 10$). The kernel function used in this experiment is a Gaussian kernel whose hyper-parameters are determined according to the maximum likelihood[1].

Fig. 3(b) shows the normalized mean squared error (nMSE) of the regression result and calculation time. The blue solid line, green solid line, red solid line and black solid line in Fig. 3(b) show the regression results of Tree-LSH-GPR-3-L, Tree-LSH-GPR-4-L, Tree-LSH-GPR-5-L and FULL-GPR, respectively.

nMSE is large when the number of hash-keys is small. However, the performance of Tree-LSH-GPR gets close to the Full-GPR by increasing the number of hash-keys. Error caused by dataset division can be reduced by increasing the number of hash-keys. The computational time of prediction by LSH-GPR-3-10 is the highest in Tree-LSH-GPR, but takes only 1/6 of that used in Full-GPR. The computational time of inverse matrix by LSH-GPR-5-1 is the lowest in Tree-LSH-GPR and becomes smaller than 1/1000 of that used in Full-GPR. The computational time of prediction is proportional to the number of hash-keys. However, the calculations can be sped up by using a parallel computing technique since each decomposed GPR can be performed independently.

4 Conclusion

In this paper, we proposed a fast approximation method for Gaussian process regression where a dataset is divided into small subsets and the outputs of the redundant number of approximated GPRs are integrated using a product-of-experts model. We applied our method to regression tasks using both artificial and real data, and showed that the proposed method is computationally efficient while the precision of the estimation is similar to previous method. In our method, the sample size of a subset affects the precision. To develop a method to determine the sample size of a subset for sufficient approximation is one of our future projects.

Sparse GPR[9][13][14][10] is a well-known fast approximation method for GPR. In this method, representative samples are extracted from the training dataset to avoid the performance degradation. When all training samples affect the entire input space, this method might work well. On the other hand, when most of the training samples affect only a small area close to themselves, wasteful calculations might emerge. This property is opposite to our method; to develop a combination of these methods is also one of our future projects.

Acknowledgments. A part of this research was supported by a contract with the Ministry of Internal Affairs and Communications entitled, 'Novel and innovative R&D making use of brain structures'.

References

1. Bishop, C.M.: Pattern recognition and machine learning, 1st edn. corr. 2nd printing edition. Springer (October 2006)
2. Bo, L., Sminchisescu, C.: Greedy block coordinate descent for large scale gaussian process regression. Computing Research Repository (2012)
3. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 253–262 (2004)
4. Foster, L., Waagen, A., Aijaz, N., Hurley, M., Luis, A., Rinsky, J., Satyavolu, C., Way, M.J., Gazis, P., Srivastava, A.: Stable and efficient gaussian process calculations. Journal of Machine Learning Research 10, 857–882 (2009)
5. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. In: Neural Computation, vol. 14, pp. 1771–1800. MIT Press (2002)
6. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613 (1998)
7. Ko, J., Fox, D.: Learning gp-bayesfilters via gaussian process latent variable models. Autonomous Robots 30, 3–23 (2011)
8. Lawrence, N.: Probabilistic non-linear principal component analysis with gaussian process latent variable models. Journal of Machine Learning Research 6, 1783–1816 (2005)
9. Lawrence, N., Seeger, M., Herbrich, R.: Fast sparse gaussian process methods: The informative vector machine. In: Advances in Neural Information Processing Systems, vol. 15, pp. 609–616 (2003)

10. Quiñonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate gaussian process regression. Journal of Machine Learning Research 6, 1939–1959 (2005)
11. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press (2006)
12. Sanjoy, D., Yoav, F.: Random projection trees and low dimensional manifolds. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 537–546 (2008)
13. Smola, A.J., Bartlett, P.: Sparse greedy gaussian process regression. In: Advances in Neural Information Processing Systems, vol. 13, pp. 619–625. MIT Press (2001)
14. Snelson, E., Ghahramani, Z.: Sparse gaussian processes using pseudo-inputs. In: Advances in Neural Information Processing Systems, vol. 18, pp. 1257–1264. MIT Press (2006)
15. Solak, E., Murray-Smith, R., Leithead, W.E., Leith, D.J., Rasmussen, C.E.: Derivative observations in gaussian process models of dynamic systems. In: Advances in Neural Information Processing Systems, vol. 15, MIT Press (2003)
16. Waterhouse, S.R.: Classification and regression using mixtures of experts. PhD thesis. Citeseer (1997)

An Analytical Approach to Single Node Delay-Coupled Reservoir Computing

Johannes Schumacher, Hazem Toutounji, and Gordon Pipa

Institute of Cognitive Science, University of Osnabrück, Germany
{joschuma,htoutounji,gpipa}@uos.de

Abstract. Reservoir computing has been successfully applied in difficult time series prediction tasks by injecting an input signal into a spatially extended reservoir of nonlinear subunits to perform history-dependent nonlinear computation. Recently, the network was replaced by a single nonlinear node, delay-coupled to itself. Instead of a spatial topology, subunits are arrayed in time along one delay span of the system. As a result, the reservoir exists only implicitly in a single delay differential equation, numerical solving of which is costly. We derive here approximate analytical equations for the reservoir by solving the underlying system explicitly. The analytical approximation represents the system accurately and yields comparable performance in reservoir benchmark tasks, while reducing computational costs by several orders of magnitude. This has important implications with respect to electronic realizations of the reservoir and opens up new possibilities for optimization and theoretical investigation.

1 Introduction

Predicting future behavior and learning temporal dependencies in time series of complex natural systems remains a major goal in many disciplines. In Reservoir Computing, the issue is tackled by projecting input time series into a recurrent network of nonlinear subunits [2, 4]: Recurrency provides memory of past inputs, while the nonlinear subunits expand their informational features. History-dependent nonlinear computations are then achieved by simple linear readouts of the network activity.

In a recent advancement, the recurrent network was replaced by a single nonlinear node delay-coupled to itself [1]. Such a setup is formalized by a delay differential equation which can be interpreted as an "infinite dimensional" dynamical system. Whereas classical reservoirs have an explicit spatial representation, a delay-coupled reservoir (DCR) uses temporally extended sampling points across the span of its delayed feedback, termed *virtual nodes*. The main advantage of such a setup is that it allows for easy realization in optical and electronic hardware [8].

A drawback of this approach is the fact that the actual reservoir computer is always only implicit in a single delay differential equation. Consequently, in many implementations the underlying system has to be solved numerically. This leads

© Springer-Verlag Berlin Heidelberg 2013

to a computational bottleneck and creates practical limitations for reservoir size and utility. The lack of reservoir equations also presents problems for applying optimization procedures.

To overcome this, we present here a recursive analytical solution used to derive approximate virtual node equations. The solution is assessed in its computational capabilities as DCR and compared against numerical solvers in nonlinear benchmark tasks. We show that while computational performance is comparable, the analytical approximation leads to considerable savings in computation time, allowing the exploration of exceedingly large setups. Finally, we discuss the perspectives of this approach regarding optimization schemes in the fashion of previous work by the authors [9].

2 Methods

2.1 Single Node Delay-Coupled Reservoirs

In a DCR, past and present information undergoes nonlinear mixing via injection into a nonlinear node with delayed feedback. Formally, these dynamics can be modeled by a delay differential equation

$$\frac{dx(t)}{dt} = -x(t) + f(x(t-\tau), J(t)), \tag{1}$$

where τ is the delay time, J is the input driving the system, and f is a nonlinear function. For a DCR, system (1) can be operated in a simple regime that is governed by a single fixed point in case $J(t) = const.$

Injecting a signal into the reservoir is achieved by multiplexing it in time: The DCR receives a single constant input $u(\bar{t})$ in each reservoir time step $\bar{t} = \lceil \frac{t}{\tau} \rceil$, corresponding to one τ-cycle of the system. During each τ-cycle, the input is again linearly transformed by a mask that is piecewise constant for short periods θ_i, representing the spacing between sampling points of $i = 1, ..., N$ virtual nodes along the delay line. Here, the mask M is chosen to be binary with random mask bits $M_i \in \{-0.1, 0.1\}$, so that node i receives a weighted input $M_i u(\bar{t})$. The masking procedure effectively prevents the driven dynamics of the underlying system from saturating. Accordingly, the sampling point spacing satisfies $\sum_{i=1}^{N} \theta_i = \tau$.

A sample is read out at the end of each θ_i, yielding N predictor variables (virtual nodes) $x_i(\bar{t})$ per time step $\bar{t}$. Computations are performed on the predictors using a linear regression model for some scalar target time series y, given by $\hat{y}(\bar{t}) = \sum_{i=1}^{N} \alpha_i x_i(\bar{t})$, where x_i, $i = 1, ..., N$ denote the DCR's virtual nodes (see eq. (4)), and the α_i are the coefficients determined by regression, e.g. using the *least squares solution* minimizing the sum of squared errors, $\sum_{\bar{t}}(y(\bar{t}) - \hat{y}(\bar{t}))^2$.

2.2 Approximate Virtual Node Equations

In the following, we discuss a recursive analytical solution to equation (1), known as *method of steps*. The resulting formulas are used to derive a piecewise solution scheme for sampling points across τ that correspond to the reservoir's virtual nodes. Finally, we use the *trapezoidal rule* for further simplification, hereby deriving approximate virtual node equations, the temporal dependencies of which only consist of other virtual nodes. As will be shown in the remainder of this article, the resulting closed-form solutions allow reservoir computation without significant loss of performance as compared to a system obtained by explicit numerical solutions, e.g. *Heun's method* ((1,2) Runge-Kutta).

First, we discuss a simple application of the *method of steps*. For better readability, the argument $J(t)$ of the nonlinearity f is omitted in this part of the derivation. If system (1) is evaluated at $(i-1)\tau \leq t \leq i\tau$ (say $t_i = i\tau$), where a continuous function $\phi_i \in C_{[(i-2)\tau,(i-1)\tau]}$ is the solution for $x(t)$ on the previous τ-interval, we can replace $x(t-\tau)$ by $\phi_i(t-\tau)$. Consequently, elementary *variation of parameters* is applicable and yields directly the solution to the initial value problem in (1) with initial value $x(t_0 = (i-1)\tau) = \phi_i(t_{i-1})$, given by

$$\begin{aligned} x(t) &= \phi_i(t_{i-1})e^{t_{i-1}-t} + e^{t_{i-1}-t}\int_{(i-1)\tau}^{t} f(\phi_i(s-\tau))e^{s-t_{i-1}}ds \\ &= \phi_i(t_{i-1})e^{t_{i-1}-t} + e^{t_{i-1}-t}\int_{(i-2)\tau}^{t-\tau} f(\phi_i(u))e^{u-(i-2)\tau}du. \end{aligned} \tag{2}$$

Further, we use the *cumulative trapezoidal rule* [5] $\int_a^b g(x)dx = \frac{h}{2}g(\chi_0 = a) + h\sum_{j=1}^{n-1} g(\chi_j) + \frac{h}{2}g(\chi_n = b)$ to interpolate the integral in (2) piece-wise linearly along a uniform grid $(i-2)\tau = \chi_0 < ... < \chi_N = t-\tau$, where $\chi_{j+1} - \chi_j = h$. With $g(\chi_j) = e^{\chi_j-(i-2)\tau}f(\phi_i(\chi_j))$, this yields

$$\begin{aligned} x(t) \approx{}& \phi_i(t_{i-1})e^{t_{i-1}-t} + e^{t_{i-1}-t}\frac{h}{2}f(\phi_i(\chi_0)) \\ &+ e^{t_{i-1}-t}\frac{h}{2}\left(e^{\chi_N-(i-2)\tau}f(\phi_i(\chi_N)) + 2\sum_{j=1}^{N-1} e^{\chi_j-(i-2)\tau}f(\phi_i(\chi_j))\right). \end{aligned} \tag{3}$$

We are now interested in $1 \leq k \leq N$ single node equations $x_k(\bar{t})$, where $\bar{t} = i$ denotes discrete reservoir time step i in case $(i-1)\tau \leq t \leq i\tau$. Assuming equidistant virtual nodes where $\tau = N\theta$ and N the number of virtual nodes, we choose a uniform grid $\chi_j = (i-2)\tau + j\theta$ with $j = 0, ..., N$ (i.e. $h = \theta$). To get an expression for $x_k(\bar{t})$, we now have to evaluate equation (3) at the sampling point $t = (i-1)\tau + k\theta$, which results in

$$\begin{aligned} x_k(\bar{t}) &= x((i-1)\tau + k\theta) \\ &\approx e^{-k\theta}\phi_i((i-2)\tau + N\theta) + \frac{\theta}{2}e^{-k\theta}f[\phi_i((i-3)\tau + N\theta)] \\ &\quad + \frac{\theta}{2}f[\phi_i((i-2)\tau + k\theta)] + \theta\sum_{j=1}^{N-1} e^{(j-k)\theta}f[\phi_i((i-2)\tau + j\theta)] \end{aligned}$$

$$
\begin{aligned}
&= e^{-k\theta} x_N(\bar{t}-1) + \frac{\theta}{2} e^{-k\theta} f[x_N(\bar{t}-2), J_N(\bar{t}-1)] \\
&\quad + \frac{\theta}{2} f[x_k(\bar{t}-1), J_k(\bar{t})] \\
&\quad + \sum_{j=1}^{k-1} \underbrace{\theta e^{(j-k)\theta}}_{c_{kj}} f[x_j(\bar{t}-1), J_j(\bar{t})]. \qquad (4)
\end{aligned}
$$

Here $J_j(\bar{t})$ denotes the masked input $M_j u(\bar{t}) \in \mathbb{R}$ (see sec. 2.1) to node j at reservoir time step $\bar{t}$, which was omitted as an argument to f during the derivation to avoid cluttering. Note that equation (4) only has dependencies on sampling points corresponding to other virtual nodes. An exemplary coupling coefficient is indicated by c_{kj}, weighting a nonlinear coupling from node j to node k. We use this to derive weight matrices that allow simultaneous computation of all nodes in one reservoir time step $\bar{t}$ by a single vector operation, hereby dramatically reducing the computation time of simulating the system by several orders of magnitude as compared to an explicit second order numerical solver.

3 Results

We compare the analytical approximation of the system, derived in the previous section, to a numerical solution obtained using Heun's method with a stepsize of 0.1. The latter is chosen due to the relatively low computational cost and provides sufficient accuracy in the context of DCR computing. As a reference for absolute accuracy, we use numerical solutions obtained with *dde23* [7], an adaptive (2,3) Runge-Kutta based method for delay differential equations. The nonlinearity f is chosen according to the Mackey-Glass equation for the remainder of this paper, such that the system is given by

$$\dot{x}(t) = -x(t) + \frac{\eta(x(t-\tau) + \gamma J(t))}{1 + (x(t-\tau) + \gamma J(t))^p}, \qquad (5)$$

where η, γ and p are metaparameters, τ the delay length, and $J(t)$ is the temporally stretched input $u(\bar{t})$, multiplexed with a binary mask M.

Note that the trapezoidal rule used in the analytical approximation, as well as Heun's method, are both second order numerical methods that should yield a global truncation error of the same complexity class. As a result, discrepancies originating from different step sizes employed in the two approaches (e.g. 0.2 in the analytical approximation and 0.1 in the numerical solution) may be remedied by simply decreasing θ in the analytical approximation, for example by increasing N while keeping a fixed τ (see sec. 3.4).

3.1 Trajectory Comparison

In a first step, we wish to establish the general accuracy of the analytical approximation in a DCR relevant setup. Figure 1 shows a comparison of reservoir

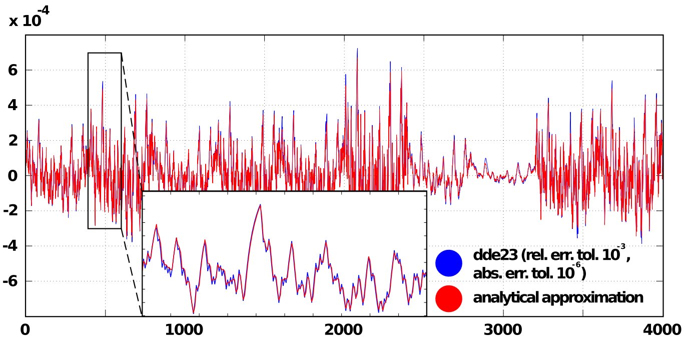


Fig. 1. Comparison between analytical approximation and numerical solution for an input-driven Mackey-Glass system with parameters $\eta = 0.4$, $\gamma = 0.05$ and $p = 1$, sampled at the temporal positions of virtual nodes, with a distance $\theta = 0.2$

trajectories computed with equation (4) (red) against trajectories computed numerically using *dde23* (blue) with relative error tolerance 10^{-3} and absolute error tolerance 10^{-6}. The systems received uniformly distributed input $u(\bar{t}) \sim \mathcal{U}_{[0,0.5]}$. The sample points correspond to the activities of $N = 400$ virtual nodes with a temporal distance of $\theta = 0.2$, and $\tau = 80$ accordingly. Given 4000 samples (corresponding to 10 reservoir time steps $\bar{t}$), the mean squared error between the trajectories is $MSE = 5.4 \times 10^{-10}$. As can be seen in the figure, the trajectories agree very well in the fixed point regime of the system (autonomous case). Although it is expected that the MSE would increase in more complex dynamic regimes (e.g. chaos), the latter are usually not very suitable for a DCR for various reasons. The following results also show a high task performance of the analytical approximation when used for DCR computing.

3.2 NARMA-10

A widely used benchmark in reservoir computing is the capacity of the DCR to model a nonlinear autoregressive moving average system y in response to uniformly distributed scalar input $u(k) \sim \mathcal{U}_{[0,0.5]}$. The NARMA-10 task requires the DCR to compute at each time step k a response

$$y(k+1) = 0.3y(k) + 0.05y(k)\sum_{i=0}^{9} y(k-i) + 1.5u(k)u(k-9) + 0.1.$$

Thus, NARMA-10 requires modeling of quadratic nonlinearities and shows a strong history dependence that challenges the DCR's memory capacity. We measure performance in this task using the correlation coefficient $r(y, \hat{y}) \in [-1, 1]$ between the target time series y and the DCR output $\hat{y}$ in response to u. Here, the DCR is trained (see sec. 2.1) on 3000 data samples, while $r(y, \hat{y})$ is computed on an independent validation data set of size 1000. Figure 2A summarizes the performance of 50 different trials for a DCR computed using the analytical approximation (see eq. 4), shown in red, as compared to a DCR simulated with Heun's method, shown in blue. Both reservoirs consist of $N = 400$ virtual nodes, evenly spaced with a distance $\theta = 0.2$ along a delay line $\tau = 80$. Both systems show a comparable performance across the 50 trials, with a median correlation coefficient between $r(y, \hat{y}) = 0.96$ and 0.97, respectively.

3.3 5-Bit Parity

As a second benchmark, we chose the delayed 5-bit parity task [6], requiring the DCR to handle binary input sequences on which strong nonlinear computations have to be performed with arbitrary history dependence. Given a random input sequence u with $u(k) \in \{-1, 1\}$, the DCR has to compute at each time step k the parity $p_m^\delta(k) = \prod_{i=0}^m u(k-i-\delta) \in \{-1,1\}$, for $\delta = 0, ..., \infty$. The *performance* ϕ_m is then calculated on n data points as $\phi_m = \sum_{\delta=0}^\infty \kappa_m^\delta$, where *Cohen's Kappa*

$$\kappa_m^\delta = \frac{\frac{1}{n}\sum_{k=1}^n \max(0, p_m^\delta(k)\hat{y}(k)) - p_c}{1 - p_c} \in \{0, 1\}$$

normalizes the average number of correct DCR output parities $\hat{y}$ by the chance level $p_c = 0.5$. We used 3000/1000 data points in training and validation set respectively. To compare performance between analytical approximation and numerical solution of the DCR, we chose $m = 5$ and truncated ϕ_m at $\delta = 7$, such that $\phi_5 \in [0, 7]$. For parameters $\eta = 0.24$, $\gamma = 0.32$ and $p = 1$, and a DCR comprised of 400 neurons ($\tau = 80$), figure 2B shows that performance ϕ_5 is comparable for both versions of the DCR, with median performances between 4.3 and 4.5. across 50 different trials of this task. As the performance is far from the ideal value of 7 and the model suffers slightly from overfitting (not shown), it is clear that the delayed 5-bit parity task is a hard problem which leaves much space for improvement.

3.4 Large Setups

We repeated the tasks in larger network setups where the computational cost of the numerical solver becomes prohibitive. In addition to increasing the number of virtual nodes N one can also decrease the node distance θ, thus fitting more nodes into the same delay span τ. Although too small θ may affect a virtual node's computation negatively, decreasing θ increases the accuracy of the analytical approximation.

NARMA-10. We illustrate this by repeating the NARMA-10 task with $N = 2000$ virtual nodes and $\tau = 200$. This results in $\theta = 0.1$, corresponding to the step size used in the numerical solution before. Note that this hardly increases the computational cost of the analytical approximation since the main simulation loop along reservoir time steps $\bar{t}$ (τ-cycles) remains unchanged. L^2-regularization is employed to manage the large number of predictors. The results are summarized for 50 trials in figure 2C (right boxplot). The median correlation coefficient

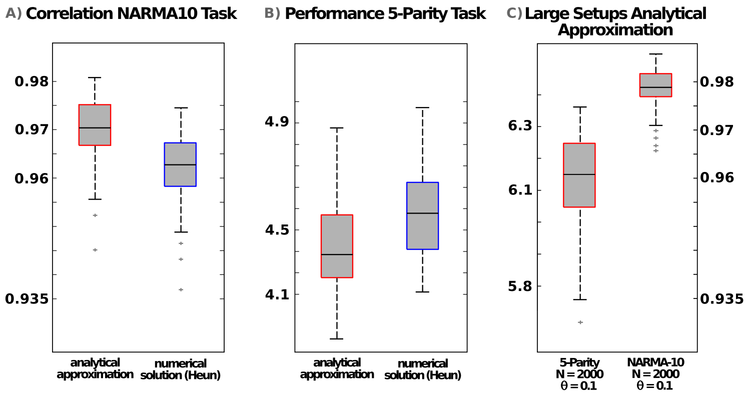


Fig. 2. Comparison on nonlinear tasks between analytical approximation and numerical solution for an input-driven Mackey-Glass system, sampled at the temporal positions of virtual nodes with a distance $\theta = 0.2$. Mackey-Glass parameters are $\eta = 0.4$, $\gamma = 0.05$ and $p = 1$ (NARMA-10) and $\eta = 0.24$, $\gamma = 0.32$ and $p = 1$ (5-bit parity), respectively. Results are reported for 400 neurons ($\tau = 80$) on data sets of size 3000/1000 (training/validation) in figures 2A and 2B, size 3000/1000 in 2C (right plot), as well as for data sets of size 10000/10000 in figure 2C (left plot). Each plot is generated from 50 different trials. The plots show median (black horizontal bar), $25^{th}/75^{th}$ percentiles (boxes), and most extreme data points not considered outliers (whiskers).

increased significantly to nearly 0.98 while the variance across trials is notably decreased (compare fig. 2A).

5-Bit Parity. For the 5-bit parity task, we addressed the task complexity by increasing both, training and validation sets, to a size of 10000. Second, we increased once more the virtual network size to $N = 2000$ virtual nodes and $\tau = 200$. The performance of the resulting DCR setup, computed across 50 trials using the analytical approximation, is summarized in figure 2C (left boxplot). The model no longer suffers as much from overfitting and the performance on the validation set increased dramatically to a median value of 6.15, which is now close to the theoretical limit of 7. While the computation to produce figure 2C took only few minutes with the analytical approximation, it is estimated that the use of the numerical solver for the same computation would have exceeded 2 days, despite the large step size of 0.1.

4 Discussion

In summary, we have developed analytical alternatives to evaluate and approximate solutions of delay differential equations that can be used for delay-coupled reservoir computing. It is shown that the resulting update equations in principle lose neither accuracy with respect to the system dynamics nor computational power in DCR benchmark tasks. Using the analytical approximation reduces computational costs considerably. This enabled us to study larger networks of delay-coupled nodes, yielding a dramatic increase in nonlinear benchmark performance. These results can lead to serious improvement regarding the implementation of DCRs on electronic boards.

Moreover, the approach yields an explicit handle on the DCR components which are otherwise implicit in equation (1). This creates new possibilities to investigate delay-coupled reservoirs and provides the basis for optimization schemes, a crucial necessity prior to any hardware implementation. Together with the reduction in computation time, this makes the use of supervised batch-update algorithms feasible to directly optimize model metaparameters (see eq. (5)) instead of conducting costly parameter scans. In addition, the optimization may include unsupervised gradient descent schemes on DCR parameters (e.g. θ, τ, N) with respect to information theoretic objectives. It is also straight forward to extend eq. (4) to account for nonuniform node spacings θ_i, subject to individual optimization (compare [9]). Continuing this line of thought, it is now possible to modify the update equations directly according to self-organizing homeostatic principles, inspired, for example, by neuronal plasticity mechanisms (e.g. [3]). We intend to explore these possibilities further in future work to maximize the system's computational power and render it adaptive to information content in task-specific setups.

References

[1] Appeltant, L., Soriano, M.C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C.R., Fischer, I.: Information processing using a single dynamical node as complex system. Nature Communications 2, 468 (2011)

[2] Herbert Jäger. The " echo state " approach to analysing and training recurrent neural networks. Technical report (2001)

[3] Lazar, A., Pipa, G., Triesch, J.: SORN: a self-organizing recurrent neural network. Frontiers in Computational Neuroscience 3 (2009)

[4] 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)

[5] Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics, 2nd edn. Texts in Applied Mathematics, vol. 37. Springer, Berlin (2006)

[6] Schrauwen, B., Buesing, L., Legenstein, R.A.: On computational power and the order-chaos phase transition in reservoir computing. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) NIPS, pp. 1425–1432. Curran Associates, Inc. (2008)

[7] Shampine, L.F., Thompson, S.: Solving ddes in matlab. In: Applied Numerical Mathematics, vol. 37, pp. 441–458 (2001)

[8] Soriano, M.C., Ortín, S., Brunner, D., Larger, L., Mirasso, C.R., Fischer, I., Pesquera, L.: Optoelectronic reservoir computing: tackling noise-induced performance degradation. Optics Express 21(1), 12–20 (2013)

[9] Toutounji, H., Schumacher, J., Pipa, G.: Optimized Temporal Multiplexing for Reservoir Computing with a Single Delay-Coupled Node. In: The 2012 International Symposium on Nonlinear Theory and its Applications, NOLTA 2012 (2012)

Applying General-Purpose Data Reduction Techniques for Fast Time Series Classification

Stefanos Ougiaroglou[1,⋆], Leonidas Karamitopoulos[2], Christos Tatoglou[2], Georgios Evangelidis[1], and Dimitris A. Dervos[2]

[1] Department of Applied Informatics, University of Macedonia, 156 Egnatia St, GR-54006 Thessaloniki, Greece
{stoug,gevan}@uom.gr

[2] Information Technology Department, Alexander TEI of Thessaloniki, GR-57400 Sindos, Thessaloniki, Greece
{lkaramit,dad}@it.teithe.gr, xtatty@gmail.com

Abstract. The one-nearest neighbour classifier is a widely-used time series classification method. However, its efficiency depends on the size of the training set as well as on data dimensionality. Although many speed-up methods for fast time series classification have been proposed, state-of-the-art, non-parametric data reduction techniques have not been exploited on time series data. This paper presents an experimental study where known prototype selection and abstraction data reduction techniques are evaluated both on original data and a dimensionally reduced representation form of the same data from seven time series datasets. The results show that data reduction, even when applied on dimensionally reduced data, can in some cases improve the accuracy and at the same time reduce the computational cost of classification.

Keywords: time series classification, nearest neighbor, data reduction.

1 Introduction

Classification methods based on similarity search have been proven to be effective approaches for time series data. More specifically, the one-Nearest Neighbour (1NN) classifier is a widely-used method. It works by assigning to an unclassified time series the class label of its most similar training time series. The main drawback of similarity-based classifiers is that all similarities between an unclassified time series item and the training time series items must be estimated. For large and high dimensional time series training sets, the high computational cost renders the application of such classifiers prohibitive. Time series classification can be sped-up using indexing, representation and/or data reduction.

Indexing can speed-up classification tremendously, but works well only in low dimensions. Thus, one must first use a dimensionality reduction technique to acquire a representation of the original data in lower dimensions. A representation may be considered as a transformation technique that maps a time series

⋆ S. Ougiaroglou is supported by the State Scholarships Foundation of Greece (I.K.Y.)

© Springer-Verlag Berlin Heidelberg 2013

from the original space to a feature space, retaining the most important features. There have been several time series representations proposed in the literature, mainly for the purpose of reducing the intrinsically high dimensionality of time series [5].

The main goal of data reduction is to reduce the computational cost of the kNN classifier and the storage requirements of the Training Set (TS). Data Reduction Techniques (DRTs)[1] try to build a small representative set of the initial training data. This set is called the Condensing Set (CS) and has the benefits of low computational cost and storage requirements while maintaining the classification accuracy at high levels. DRTs can be divided into two algorithm categories: (i) Prototype Selection (PS) [6], and, (ii) Prototype Abstraction (PA) (or generation) [11]. Although both categories have the same motivation, they differ on the way they build the CS. PS algorithms select some TS items and use them as representatives, whereas, PA algorithms generate representative items by summarizing similar TS items.

Data reduction has been recently exploited for fast time series classification. More specifically, [3] and [12] propose PS algorithms for speeding-up 1NN time series classification. The disadvantage of these methods is that they are parametric. The user must define the CS size through trial-and-error procedures.

The present work has been motivated by the following observations.To the best of our knowledge, state-of-the-art non-parametric PS and PA algorithms have not been evaluated neither on original time series nor on their reduced dimensionality representations. Also, a PA algorithm we have previously proposed (RHC [9]) has not been evaluated on time series data. The contribution of this paper is the experimental evaluation of two PS algorithms, namely, CNN-rule [7] and IB2 [2,1], and two PA algorithms, namely, RSP3 [10] and RHC [9] both on original time series data and a reduced dimensionality representation of the same data. Our study adopts the Piecewise Aggregate Approximation (PAA) [8,13] time series representation method. The goal is to test how classification is affected when applying data reduction on dimensionally reduced time series. PAA is a very simple dimensionality reduction technique that segments a time series into h consecutive sections of equal-width and calculates the corresponding mean for each one. The series of these means is the new representation of the original data.

The rest of the paper is organized as follows. Section 2 discusses the details of the four aforementioned DRTs. Section 3 describes the experimental study and the obtained results, and Section 4 concludes the paper.

2 Data Reduction Techniques

In this section, we present the four DRTs we use in our experimentation. They are based on a simple idea: Data items that do not define decision boundaries between classes are useless for the classification process. Therefore, they can be

[1] One can claim that dimensionality reduction is also data reduction. However, we consider DRTs only from the item reduction point of view.

discarded. DRTs try to select or generate a sufficient number of items that lie in data areas close to decision boundaries. The DRTs we deal with in this Section are non-parametric. They automatically determine the size of CS based on the level of noise and the number of classes in the data (the more the classes, the more boundaries exist and, thus, the more items get selected or generated).

2.1 Prototype Selection Algorithms

Hart's Condensing Nearest Neighbour Rule (CNN-Rule). The CNN-rule [7] is the earliest and the best known PS algorithm. It uses two sets, S and T. Initially, a TS item is placed in S, while all the other TS items are placed in T. Then, CNN-rule tries to classify the content of T by using the 1NN classifier on the content of S. When an item is misclassified, it is considered to lie in a data area close to decision boundaries. Thus, it is transferred from T to S. The algorithm terminates when there are no transfers from T to S during a complete pass of T. The final instance of set S constitutes the CS. The multiple passes on data ensure that the remaining items in T are correctly classified by applying the 1NN classifier on CS. The algorithm is based on the following simple idea: items that are correctly classified by 1NN, are considered to lie in a central-class data area and thus, they are ignored. In contrast, items that are misclassified, are considered to lie in a close-class-border data area, and thus, they are placed in CS. The weak point of the CNN-rule is that the resulting CS depends on the order of items in TS. This means that different CSs are build by examining the same data in a different order.

IB2 Algorithm. IB2 belongs to the well-known family of IBL algorithms [2,1]. It is based on CNN-rule. Actually, IB2 is a simple one pass variation of CNN-rule. Each TS item x is classified using 1NN on the current CS. If x is classified correctly, it is discarded. Otherwise, x is transferred to CS. Contrary to CNN-rule, IB2 does not ensure that all discarded items can be correctly classified by the final content of CS. However, since it is a one-pass algorithm, it is very fast. In addition, IB2 builds its CS incrementally. New TS items can be taken into consideration after the CS creation. Thus, IB2 is appropriate for dynamic (streaming) environments where new TS items may gradually arrive. Also, contrary to CNN-rule and many other DRTs, IB2 does not require that all TS data reside into the main memory. Therefore, it can be applied in devices whose memory is insufficient for storing all the TS data. Like CNN-rule, IB2 is a data order dependent algorithm.

2.2 Prototype Abstraction Algorithms

RSP3 Algorithm. The RSP3 algorithm belongs to the popular family of Reduction by Space Partitioning (RSP) algorithms [10]. This family includes three PA algorithms. All of them are based on the idea of the early PA algorithm of Chen and Jozwik (CJ algorithm) [4] that works as follows: First, it retrieves the most distant items A and B of TS that define its diameter. Then, it divides the

TS into two sets, S_A and S_B. S_A includes TS items that are closer to A, while S_B includes TS items that are closer to B. CJ proceeds by selecting to divide the set with the larger diameter. This procedure continues until the number of sets becomes equal to a user defined number. In the end, for each set S, CJ averages the items that belong to the most common class in S and creates a mean item. The created mean items constitute the final CS.

RSP1 is a simple variation of CJ that for each final set creates as many mean items as the number of distinct classes in the set. RSP2 differs on how it selects the next set that will be divided. Instead of the criterion of the largest diameter, RSP2 uses the criterion of overlapping degree. RSP3 is based on the concept of homogeneity. A set is homogeneous when it includes items of only a specific class. The algorithm continues dividing the created sets until all of them became homogeneous. Considering RSP3, we observe that the algorithm generates more prototypes for the close borders data areas and fewer for the "central" data areas. RSP3 is the only non-parametric algorithm of the RSP family (CJ included). All these algorithms do not depend on the ordering of the data items in TS.

Reduction through Homogeneous Clusters (RHC) Algorithm. RHC [9] is also based on the concept of homogeneity. Initially, the whole TS is considered as a non-homogeneous cluster C. RHC begins by computing a mean item for each class (class centroid) in C. Then, it applies k-means clustering on C using the class centroids as initial means. The clustering procedure builds as many clusters as the number of classes in C. The aforementioned clustering procedure is applied recursively on each non-homogeneous cluster. In the end, the centroids of the homogeneous clusters are stored in CS. By using the class centroids as initial means for the k-means clustering, the algorithm attempts to quickly find homogeneous clusters and achieve high reduction rates. RHC is independent on the ordering of data in TS. The results of the experimental study in [9] show that RHC achieves higher reduction rates (smaller CSs) and is faster than RSP3 and CNN-rule, while the classification accuracy remains at high levels.

3 Experimental Study

3.1 Experimental Setup

The four presented DRTs were evaluated on seven time series datasets distributed by the UCR time-series classification/clustering website[2]. Table 1 summarizes the datasets used. All datasets are available in a training/testing form. We merged the training and testing parts and then we randomized the resulting datasets. No other data transformation was performed. All algorithms were coded in C and as a similarity measure we used the Euclidean distance.

[2] `http://www.cs.ucr.edu/~eamonn/time_series_data/`

Table 1. Time-series datasets description

Time-series dataset	Size (time-series)	Length (Attr.)	Classes
Synthetic Control (SC)	600	60	6
Face All (FA)	2250	131	14
Two-Patterns (TP)	5000	128	4
Yoga (YG)	3300	426	2
Wafer (WF)	7164	152	2
Sweadish Leaf (SL)	1125	128	15
CBF	930	128	3

We report on the experiment we conducted with a certain value for the parameter of the PAA representation due to space limitations. We applied the PAA representation on time series by setting the number of dimensions equal to twelve (h=12). Most of the research work provides experimental results with values of h ranging from 2 to 20. We found that lower values of h have a negative effect on the classification accuracy, whereas higher values give time series that cannot be efficiently indexed by multi-dimensional indexing methods. In our future work, we plan to further investigate the effect the dimensionality of time series has on the performance of classification.

All experiments were run twice, once on the original time series and once on their 12-dimensional representations. We wanted to test how the combination of data reduction and dimensionality reduction affects the performance of 1NN classification.

We evaluated the four DRTs by estimating four measurements, namely, accuracy (ACC), classification cost (CC), reduction rate (RR), and, preprocessing cost (PC). Cost measurements were estimated by counting the distance computations multiplied by the number of time series attributes (time series length). Of course, the RR and CC measurements relate to each other: the lower the RR, the higher the CC. However, CC measurements can express the cost introduced by the data dimensionality. We report the average values of these measurements obtained via five-cross-fold validation.

3.2 Comparisons

Table 2 presents the experimental results. The table includes two parts, one for the original datasets and one for the datasets obtained after applying PAA on them. Both table parts include the measurements obtained by applying the 1NN classifier on the non-reduced data (conventional 1NN). Each table cell includes the four measurements obtained by first applying a DRT on the original or 12-dimensional time series datasets (preprocessing step) and then by using 1NN on the resulting CS (classification step). The cost measurements are in million (M) distance computations. The PC measurements do not include the small cost overhead introduced by PAA.

Table 2. Experimental results on accuracy, classification cost, reduction rate and preprocessing cost

Dataset		Original dimensionality					12 dimensions				
		Conv. 1NN	**CNN**	**IB2**	**RSP3**	**RHC**	**Conv. 1NN**	**CNN**	**IB2**	**RSP3**	**RHC**
SC	Acc (%):	91.67	90.17	89.00	98.33	98.67	98.50	97.00	95.83	**98.83**	98.17
	CC (M):	3.46	0.67	0.53	1.38	0.09	0.69	0.06	0.05	0.12	**0.03**
	RR (%):	-	80.50	84.67	60.08	**97.29**	-	90.75	93.13	82.96	95.75
	PC (M):	-	7.77	1.31	16.22	2.39	-	0.89	**0.13**	3.45	0.52
FA	Acc (%):	95.07	91.60	91.02	**95.46**	93.02	87.91	83.78	82.31	87.07	84.49
	CC (M):	106.11	19.87	18.38	51.65	12.93	9.72	2.89	2.53	4.80	**2.08**
	RR (%):	-	81.28	82.68	51.32	**87.81**	-	70.23	74.01	50.58	78.59
	PC (M):	-	216.36	48.96	533.70	140.41	-	30.36	**5.95**	50.91	13.16
TP	Acc (%):	98.50	94.68	93.60	**98.10**	93.72	97.56	93.52	91.38	96.66	94.34
	CC (M):	512.00	85.66	76.83	243.51	55.50	48.00	8.22	6.86	20.42	**6.69**
	RR (%):	-	83.27	85.00	52.44	**89.16**	-	82.89	85.72	57.45	86.06
	PC (M):	-	1.169.75	205.95	2085.42	150.49	-	103.86	**17.34**	196.00	17.63
YG	Acc (%):	93.76	91.58	89.55	**92.85**	90.94	92.36	90.39	88.03	91.03	90.03
	CC (M):	742.26	138.56	108.92	229.82	93.85	20.91	4.41	3.50	6.71	**3.13**
	RR (%):	-	81.33	85.33	69.04	**87.36**	-	78.91	83.26	67.90	85.02
	PC (M):	-	1854.74	254.41	4072.30	162.61	-	52.23	8.04	110.56	**4.26**
WF	Acc (%):	99.87	99.69	99.62	**99.82**	99.55	99.79	99.62	99.51	99.40	99.25
	CC (M):	1248.30	13.59	11.72	26.88	9.37	98.55	1.21	**1.01**	1.86	**1.01**
	RR (%):	-	98.91	99.06	97.85	**99.25**	-	98.77	98.97	98.11	98.97
	PC (M):	-	165.88	31.42	7196.75	63.69	-	15.63	**2.57**	495.63	4.64
SL	Acc (%):	52.36	49.87	48.18	52.00	**52.80**	52.62	49.07	48.62	51.20	51.20
	CC (M):	25.92	15.94	14.80	19.00	12.80	2.43	1.54	1.37	1.78	**1.32**
	RR (%):	-	38.51	42.89	26.69	**50.60**	-	36.76	43.67	26.69	45.69
	PC (M):	-	112.17	31.39	1537.07	57.01	-	11.33	**2.86**	56.00	4.99
CBF	Acc (%):	98.39	98.17	97.63	**99.78**	98.60	100.00	99.57	99.35	99.68	99.57
	CC (M):	17.71	1.29	1.15	1.97	0.40	1.66	0.06	0.06	0.12	**0.04**
	RR (%):	-	92.74	93.49	88.87	**97.74**	-	96.34	96.56	92.63	97.47
	PC (M):	-	15.06	3.50	78.48	7.26	-	0.66	**0.19**	7.32	0.70
Avg	Acc (%):	89.94	87.97	86.94	90.91	89.62	89.82	87.57	86.43	89.12	88.15
	CC (M):	379.40	39.37	33.19	82.03	26.42	25.99	2.63	2.20	5.12	2.04
	RR (%):	-	79.51	81.87	63.76	87.03	-	79.24	82.19	68.05	83.94
	PC (M):	-	505.96	82.42	2217.13	83.37	-	30.71	5.30	131.44	6.56

At a glance, we observe that 1NN classification on the 12-dimensional datasets is very fast. In most cases, the preprocessing and classification cost are extremely low, while classification accuracy remains at high, acceptable levels. Therefore, we conclude that one can obtain efficient time series classifiers by combining prototype selection or abstraction algorithms with time-series dimensionality reduction representations.

It is worth mentioning that, in three datasets, the two PA algorithms, RSP3 and RHC, achieved higher classification accuracy than the conv-1NN. In the case of SC dataset, the accuracy improvements were very high. Almost in all cases, RSP3 achieved the highest accuracy. However, it is the slowest method in terms of both preprocessing and classification (RSP3 had the lowest reduction rates). The high PC measurements are attributed to the costly procedure for finding the most distant items in each created subset (see Subsection 2.2 or [10] for details).

RHC and IB2 had much lower preprocessing cost than the other two methods. This happened because IB2 is a one-pass algorithm and RHC is based on a version of k-Means that is sped-up by the class centroid initializations (see Subsection 2.2 or [9] for details). In addition, RHC builds the smallest CSs. In all cases, RHC achieved higher reduction rates than the other DRTs. Thus, the corresponding classifiers had the lowest classification costs. The classification accuracy achieved by RHC was usually higher than IB2 and CNN-rule. In some cases, RHC achieved accuracy even higher than RSP3. Considering the above, one may safely conclude that RHC is an efficient speed-up method that can deal with all comparison criteria.

No DRT can be considered as the best speed-up choice. If classification accuracy is the most critical criterion, RSP3 may be preferable. On the other hand, if fast classification and/or fast construction of the CS are more critical than accuracy, RHC may be a better choice.

4 Conclusions

Fast time series classification is a crucial data mining issue. This paper proposed the use of non-parametric state-of-the-art prototype selection and abstraction algorithms for fast time series classification.

The experimental study has shown that by combining prototype selection or abstraction algorithms with dimensionality reduction, one can obtain accurate and very fast time series classifiers. In addition, our study has shown that the abstraction algorithms can achieve even higher accuracy than the conventional 1NN classifier.

References

1. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. Int. J. Man-Mach. Stud. 36(2), 267–287 (1992)
2. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (1991)

3. Buza, K., Nanopoulos, A., Schmidt-Thieme, L.: INSIGHT: Efficient and effective instance selection for time-series classification. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 149–160. Springer, Heidelberg (2011)
4. Chen, C.H., Jóźwik, A.: A sample set condensation algorithm for the class sensitive artificial neural network. Pattern Recogn. Lett. 17, 819–823 (1996)
5. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. VLDB Endow. 1(2), 1542–1552 (2008)
6. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. 34(3), 417–435 (2012)
7. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14(3), 515–516 (1968)
8. Keogh, E.J., Pazzani, M.J.: A simple dimensionality reduction technique for fast similarity search in large time series databases. In: Terano, T., Liu, H., Chen, A.L.P. (eds.) PAKDD 2000. LNCS, vol. 1805, pp. 122–133. Springer, Heidelberg (2000)
9. Ougiaroglou, S., Evangelidis, G.: Efficient dataset size reduction by finding homogeneous clusters. In: Proceedings of the Fifth Balkan Conference in Informatics, BCI 2012, pp. 168–173. ACM, New York (2012)
10. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition 37(7), 1561–1564 (2004)
11. Triguero, I., Derrac, J., Francisco Herrera, S.G.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C 42(1), 86–100 (2012)
12. Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 1033–1040. ACM, New York (2006)
13. Yi, B.K., Faloutsos, C.: Fast time sequence indexing for arbitrary lp norms. In: Proceedings of the 26th International Conference on Very Large Data Bases, VLDB 2000, pp. 385–394. Morgan Kaufmann Publishers Inc., San Francisco (2000)

Two-Layer Vector Perceptron

Vladimir Kryzhanovsky[1], Irina Zhelavskaya[1], and Juan Antonio Clares Tomas[2]

[1] Center of Optical Neural Technologies of Scientific Research Institute for System Analysis, Russian Academy of Sciences, Vavilova st., 44/2, 119333 Moscow, Russia
[2] Institute of Secondary Education: IES SANJE, Alcantarilla, Murcia, Spain
Vladimir.Krizhanovsky@gmail.com, winjei@ya.ru, juanantonio.clares@murciaeduca.es

Abstract. A new model – two-layer vector perceptron – is offered. Though, comparing with a single-layer perceptron, its operation needs slightly more (by 5%) calculations and more effective computer memory, it excels in a much lower error rate (four orders of magnitude as lower).

Keywords: vector neural networks, Potts model.

1 Introduction

The Potts model [1-2] is the first and most-known vector neural net. The model still draws much attention of researchers from such fields as physics, medicine, image segmentation and neural networks. Later the parametric neural net [3] was offered and thoroughly studied by a small team of the Institute of Optical Neural Technologies of RAS (the Center of Optical Neural Technologies of the System Research Institute of RAS today). A similar model (CMM) was developed independently and is still investigated at York University [4]. V.M. Kryzhanovsky's thesis introduces a vector neural net model with a proximity measure between neuron states. This kind of neural net generalizes all above-mentioned models. Researchers studied both fully connected and perceptron-like architectures. Various vector-net learning rules were studied [6]. The results prove the high efficiency of vector nets.

The perceptron is most suitable for associative memory-based applications (in our case this is the vector perceptron). However, it has a major drawback: even one output neuron taking a wrong state results in an input vector not being recognized. To overcome this, one has to raise the reliability of each neuron by increasing the net redundancy or decreasing the loading of the net. Putting it in other words, the vector perceptron consists of "reliable" neurons that can't make mistakes, which contradicts the whole philosophy of the neural net.

The alternative approach is to use weak neurons. With similar requirements for RAM, a collection of weak neurons proves more effective than a small number of reliable neurons. The trick is to supply the vector perceptron with an additional layer of one neuron the number of states of which is equal to the number of stored patterns. Its aim is to accumulate the information from the preceding layer and to identify the input pattern. The approach is close to the idea offered in papers [7, 8].

© Springer-Verlag Berlin Heidelberg 2013

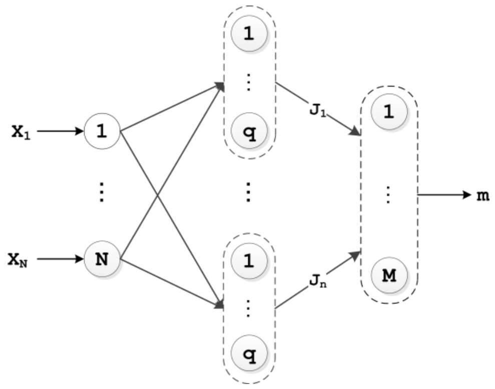


Fig. 1. The general arrangement of the double-layer vector perceptron

Ideologically, the paper consists of three parts: formal description of the model, qualitative description with a simple example that helps to understand the point of the approach, experimental results. Due to the limitation on the printed material, the authors give only most interesting experimental results.

2 Setting the Problem

Let us consider a q-digit vector **X** (its components can take one of q discrete values) which is the result of distortion of one of reference vectors ($q \geq 2$). The goal is to build a system that allows us to find reference vector $\mathbf{X}_m$ whose Hemming distance to input vector **X** is smallest. For simplicity let us turn to binary reference vectors (i.e. $q = 2$) for which the distortion is just a swapping of the components.

3 Formal Description of the Model

3.1 Describing the Model

Let us consider a double-layer architecture (Fig. 1). The input layer has N scalar neurons each of which can take two states $x_i = \pm 1$, $i=1,2,\ldots,N$. The first (inner) layer consists of n vector neurons. Each of these has $2q$ fictive states and is described by basis vectors of a q-dimensional space $\vec{\mathbf{y}}_i \in \{\pm\vec{\mathbf{e}}_1, \pm\vec{\mathbf{e}}_2, \ldots, \pm\vec{\mathbf{e}}_q\}$, where $\vec{\mathbf{e}}_k = (0,\ldots,0,1,0,\ldots,0)$ is the unit vector whose k-th digit is unit. The states being fictive means that when learnt, the neurons of the second layer have $2q$ discrete states, and in operation the neurons are regarded as simple adders. This is done to simplify the description of the model. The second (output) layer has one vector neuron which can take on M states and is described by basis vectors of the M-dimensional space $\vec{\mathbf{O}} \in \{\vec{\mathbf{o}}_1, \vec{\mathbf{o}}_2, \ldots, \vec{\mathbf{o}}_M\}$.

The state of the perceptron can be described by three vectors:

1) the input layer is described by a N-dimensional binary vector $\vec{\mathbf{X}} = (x_1, x_2, ..., x_N)$ where $x_i = \pm 1$;

2) the first (inner) layer by n-dimensional $2q$-digit vector $\vec{\mathbf{Y}} = (\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, ..., \vec{\mathbf{y}}_n)$, where $\vec{\mathbf{y}}_i \in \{\pm\vec{\mathbf{e}}_1, \pm\vec{\mathbf{e}}_2, ..., \pm\vec{\mathbf{e}}_q\}$, $\vec{\mathbf{e}}_k = (0,...,0,1,0,...,0)$ is the q-dimensional unit vector whose k-th digit is unit;

3) the second (output) layer by M-digit vector $\vec{\mathbf{O}} \in \{\vec{\mathbf{o}}_1, \vec{\mathbf{o}}_2, ..., \vec{\mathbf{o}}_M\}$, where $\vec{\mathbf{o}}_r = (0,...,0,1,0,...,0)$ is the M-dimensional unit vector holding unit in the r-th digit.

Each reference pattern $\mathbf{X}_m$ is associated uniquely to vector $\mathbf{Y}_m$. In turn, each vector $\mathbf{Y}_m$ is associated uniquely to vector $\mathbf{o}_m$. Each component of vector $\mathbf{Y}_m$ is generated so that on the one hand $\mathbf{Y}_m$ is a unique vector, and on the other hand possible states $\{\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2, ..., \vec{\mathbf{e}}_q\}$ are distributed evenly among reference vectors, i.e. $\sum_\mu \vec{\mathbf{y}}_{\mu i} \equiv \frac{M}{q}(1,1,...,1)$. If the last condition is not satisfied, the error rate grows by several orders of magnitude. So we build a neural net that stores association:

$$\vec{\mathbf{X}}_m \Leftrightarrow \vec{\mathbf{Y}}_m \Leftrightarrow \vec{\mathbf{o}}_m . \tag{1}$$

3.2 Learning Procedure

The synaptic connections of the vector perceptron are computed using generalized Hebb's rule:

$$\vec{\mathbf{W}}_{ji} = \sum_{m=1}^{M} \vec{\mathbf{y}}_j^m x_i^m \quad \text{and} \quad \hat{\mathbf{J}}_j = \sum_{m=1}^{M} \vec{\mathbf{o}}_m^T \vec{\mathbf{y}}_j^m , \tag{2}$$

where $\mathbf{W}_{ji}$ is the q-dimensional vector describing the connection between the i-th neuron of the input layer and the j-th neuron of the second layer; $\mathbf{J}_j$ is the $M \times q$ matrix responsible for the connection between the j-th neuron of the second layer and the sole output neuron; $i = \overline{1,N}$, $j = \overline{1,n}$.

3.3 Identification Process

Let a vector $\mathbf{X}$ arrive at the network input. Let us compute the response of the net $\mathbf{O}$. To this end let us first compute the local fields around the second-layer neurons:

$$\vec{\mathbf{h}}_j = \sum_{i=1}^{N} \vec{\mathbf{W}}_{ji} x_i \ . \tag{3}$$

Since the second-layer neurons act as simple adders during recognition, the signal $\mathbf{h}_j$ goes to the output neuron without any changes. That is why the local field around the output neuron has the form:

$$\vec{\mathbf{H}} = \sum_{j=1}^{n} \hat{\mathbf{J}}_j \vec{\mathbf{h}}_j^T . \tag{4}$$

The final output **O** is computed in the following manner. The number of the greatest component of the local field **H** is determined. Let it be number *r*. Then the output of the perceptron is $\mathbf{O} = \mathbf{o}_r$, in other words, the input of the perceptron receives the distorted variant of the *r*-th reference pattern. And the greater $(\mathbf{H},\mathbf{o}_r)$ is, the statistically more reliable the response of the net is. Moreover, if we arrange the numbers of the components in the increasing order, the resulting list will tell how input vector **X** is close to corresponding vectors in terms of Hemming vicinity.

4 Qualitative Description of the Model

4.1 The General Idea

Each vector neuron corresponds to a unique partition of the whole set of reference patterns into *q* subsets. For instance, Fig. 2 shows two partition of the whole set into *q*=4 subsets (*M*=12). For either partition we can calculate *q* "probabilities" (the components of the vector of local fields h_{jm}^{i}) of the input pattern belonging to one of *q* subsets. Each vector neuron is in fact a kind of solver that picks a subset of highest "probability" (in Fig. 3 it is subset No.1 in the first partition and subset No.1 in the second partition). The intersection of the subsets chosen by all solvers determines the output of a single-layer perceptron. Calculation of the "probabilities" may be accompanied by errors caused by the statistical nature of the calculation. So, a solution found by picking "highest-probability" subsets may be untrue. Mistaking a "winning" subset in at least one partition is enough to get an incorrect solution (Fig. 3).

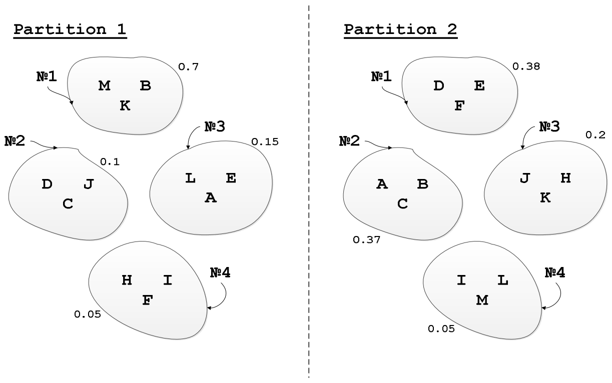


Fig. 2. Partition of a set of objects in two different ways

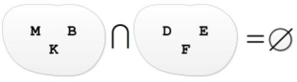


Fig. 3. Intersection of winning subsets from partition 1 and 2 results in a null subset

The goal of the method we offer is to overcome this drawback. The idea is to take interim decisions by accumulating probability information over all partitions rather than using only "probabilities" of one partition (and cutting off possible solutions by doing so). The "probabilities" obtained for each subset of all partitions will be regarded as statistical indicators of any pattern from a subset A_i^j of particular partition A^j matching input pattern **X**. After "probabilities" are evaluated for all partitions, each pattern $\mathbf{X}_m$ can be associated with a set of such statistical indicators – one from each partition:

$$\vec{\mathbf{X}}_m \leftarrow h_{jm}^i, \qquad \vec{\mathbf{X}}_m \in A_i^j, \quad j=\overline{1,n}, \; i=\overline{1,q}. \quad . \tag{5}$$

(*It should be noted once again that here the "probability" is understood as a certain statistical quantity – a component of local field* h_{jm}^i, *to be exact. The higher the probability of an input pattern being a pattern from a subset corresponding to this local field, the greater the amplitude of the component is.*)

4.2 Example

Let us exemplify the idea. Fig. 2 shows two different partitions of one set of 12 letter-denoted patterns into 4 subsets (n = 2). Let distorted pattern B arrive at the input. In the figure each subset goes with a number which is the computed "probability" of the input pattern being a pattern of this particular subset.

Table 1. Probability that the input pattern belongs to a particular subset

Partition 1			Partition 2		
Subset number	Objects	Probability*	Subset number	Objects	Probability*
1	**M, K, B**	**0.70**	**1**	**D,E,F**	**0.38**
2	**D, J, C**	0.10	**2**	**A, B, C**	0.37
3	**L, E, A**	0.15	**3**	**J, H, K**	0.20
4	**H, I, F**	0.05	**4**	**I, L, M**	0.05

*Probability - chances that the input pattern belongs to the subset.

When a single-layer perceptron is used for recognition, subset No.1 is the "winning" subset in the first partition, and it really holds the input pattern. In the second partition the "winner" is also subset No.1, yet it does not have the input pattern. The intersection of the two subsets gives a null subset (Fig. 3), which means that the net can't identify the input pattern. Clear that the failure of one neuron causes the failure of the whole system. At the same time, we see that for the second partition the probabilities of the input pattern belonging to subset 1 or subset 2 are almost equal – the difference is just 0.01 (1%) (Table 1). That is to say, it is almost equiprobable for the input pattern to belong to either the first or second subset. Our model takes this fact into account, and for each pattern the decision is made by using probabilities from both partitions (Table 2). The pattern that corresponds to the greatest summary "probability" is chosen as the response of the system. The result is a correct identification of the input pattern by the net.

Table 2. Recognition probabilities computed for either partition and their sum for each pattern

Pattern	Probability for partition 1	Probability for partition 2	Summary probability
A	0.15	0.37	0.52
B	0.70	0.37	**1.07**
C	0.10	0.37	0.47
D	0.10	0.38	0.48
E	0.15	0.38	0.53
F	0.05	0.38	0.43

5 Details of the Algorithm

	Single layer	Two layers	Ratio*
Computational burden (number of operations)	$2Nnq$	$2Nnq+(n+1)M$	1.025
Necessary amount of RAM, bytes	$4Nnq$	$4Nnq+4nM$	1.033

* - the ratio is taken for $M = 100$; $N = 100$; $q = 300$; $n = 2$.

It is seen from the table that the algorithm we offer requires just 4% more computational resources (CPU, RAM) than the single-layer perceptron.

6 Experimental Results

This section deals with a computer simulation of single- and double-layer perceptron operation and gives the comparison of the results. In Figure 4 and 5 the Y-axis of the plots is the recognition failure probability P (the perceptron fails to recognize a distorted reference vector). In both Figures the curves corresponding to the single-layer perceptron are drawn in thin line with rhombic marks (the curves are above the others). The other curves refer to the double-layer perceptron. The plots are drawn for different n and q.

If the number of patterns M, their dimensionality N and noise parameter a (the probability of a component of an input binary vector being distorted) are determined by the conditions of a problem to be solved, the number of q-digit neurons of the inner layer and the number of their states can be varied to get satisfactory reliability.

Let us first consider how the recognition failure probability varies with M and N given constant n and q (Fig. 4a and 4b). As expected, the growing dimensionality N of stored patterns or a decrease of their number M result in an exponential fall of probability P. It is also seen that the introduction of another layer allows a more than an order of magnitude (two orders and over) decrease of P. The lower the probability P for the original single-layer net, the more significantly P falls for the double-layer system.

The noise-resistance of the double-layer net is also higher – the rhomb-marked curve lies noticeably higher than the other curve (Fig. 5a, parameters M=1000, N=100, q=200).

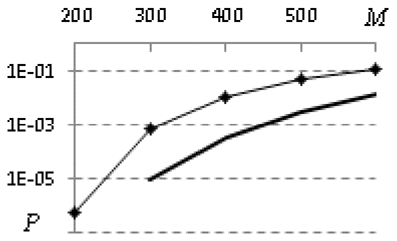


Fig. 4a. Probability P versus the number of stored patterns M. Parameters N=100, q=100, n=2.

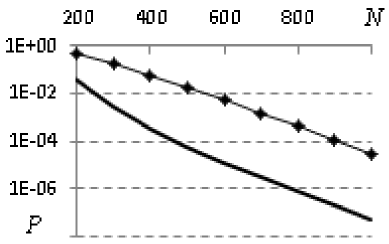


Fig. 4b. Probability P versus dimensionality N. Parameters M=1000, q=50, n=3

Fig. 5a shows a few dependences of two-layer net error probability P on noise level a for different combinations of n and q (given nq = const). The upper dashed curve refers to n=40, q=10, the curve below corresponds to n=8 and q=50. Still lower goes the curve for n=4 and q=100. The combination of n=2 and q=200 (thick solid line) demonstrates the lowest P. So we see that from the reliability viewpoint it is better to use a small number of reliable (redundant) neurons for the two-layer system. However, this sort of networks can't boast of high resistance to a failure of the net itself. The data (dashed line) shown in Fig. 5a proves that reliable and failure-resistant neural systems can be made up of unreliable elements having a considerable parameter spread.

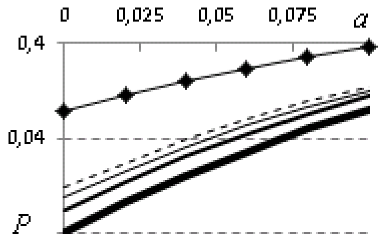


Fig. 5a. Recognition failure probability P versus noise level a. M=1000, N=100

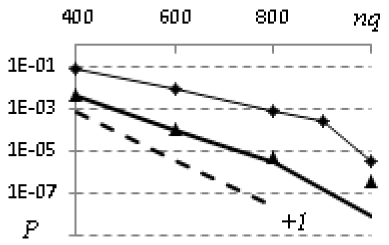


Fig. 5b. Recognition failure probability P versus nq. M=1000, N=100, a=0

The net with n=40 and q=10 differs from the net with n=2 and q=200 by the principles securing correct recognition. In the first case the second layer accumulating information from a large number of unreliable elements plays the key role (for a single-layer perceptron with given parameters the recognition probability is zero). In the latter case the second layer corrects the errors of the first layer only occasionally (thin marked line in Fig. 5a).

Fig. 5b shows how P depends on inner-layer parameters n and q. The thick line corresponds to probability P of a two-layer net with n=2 and q=200÷500, and n =2÷5 and q=200 (triangular marks). The both nets have the same computational burden and requirements for RAM. The simulation shows that 1) the growth of both parameters leads to an exponential decrease of P; 2) the both nets has the same probability P for nq < 800 (an unexpected enough result), which once again says for the conclusion drawn above.

The algorithm has yet another useful property, which a single-layer perceptron does not have. If we arrange patterns in decreasing order according to the components of their local field **H** (table 2, column “sum”), the order will tell us how close a pattern is to the input vector, a pattern in the first place being regarded as the response of the system. However, we can raise the reliability of the system still more if we compute scalar products of the input vector and first K patterns and define the final response by the maximum of this product. In Fig. 5b the dashed line corresponds to the results of simulation for K = 2. Clear that the trick allows us to further decrease P by another two orders of magnitude (see $nq = 800$).

7 Conclusion

In this work we investigate how properties of one layer vector perceptron change if an additional layer is overbuilt to perceptron output. This layer is added to accumulate statistical information about each pattern separately, i.e. to calculate estimations of Hemming distance between input vector and each pattern. This gives us several advantages. Firstly, it is the compensation statistical overswings which lead to incorrect work of one layer networks. Indeed, using as a system answer the pattern with minimum distance in computer simulations shows that the error probability decreases over than 3 orders of magnitude. Secondly, the additional layer allows us to select a few likely candidates with smallest distance estimations and then directly compare them with the input vector. In this case, computation complexity is negligibly increased, at the same time the reliability increases essentially. For example, choosing two best candidates and direct calculating Hemming distances decrease the error probability over than 4 orders of magnitude.

The research is supported by projects ONIT RAN 1.8 and 2.1.

References

1. Wu, F.Y.: The Potts model. Review of Modern Physics 54, 235–268 (1982)
2. Kanter, I.: Potts-glass models of neural networks. Physical Review A 37(7), 2739–2742 (1988)
3. Kryzhanovsky, B., Mikaelyan, A.: Doklady Mathematics. On the Recognition Ability of a Neural Network on Neurons with Parametric Transformation of Frequencies 65(2), 286–288 (2002)
4. Austin, J., Turner, A., Lees, K.: Chemical Structure Matching Using Correlation Matrix Memories. In: International Conference on Artificial Neural Networks, IEE Conference Publication 470, Edinburgh, UK, September 7-10. published by IEE, London (1999)
5. Kryzhanovsky, V.M.: Ph.D. Thesis, Research into Binary-Synaptic-Coefficient Vector Neural Nets for Data Processing and Decision Making Problems, System Research Institute of the Russian Academy of Sciences (2010)
6. Kryzhanovskiy, V., Zhelavskaya, I., Fonarev, A.: Vector Perceptron Learning Algorithm Using Linear Programming. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 197–204. Springer, Heidelberg (2012)
7. Podolak, I.T., Biel, S., Bobrowski, M.: Hierarchical classifier. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 591–598. Springer, Heidelberg (2006)
8. Podolak, I.T.: Hierarchical classifier with overlapping class groups. Expert Systems with Applications 34(1), 673–682 (2008)

Local Detection of Communities by Neural-Network Dynamics

Hiroshi Okamoto[1,2]

[1] Research & Development Group, Fuji Xerox Co., Ltd.
6-1 Minatomirai, Nishi-ku, Yokohama-shi, Kanagawa 220-8668, Japan
[2] RIKEN Brain Science Institute
2-1 Hirosawa, Wako, Saitama 351-0198, Japan
hiroshi.okamoto@fujixerox.co.jp

Abstract. Community structure is a hallmark of a variety of real-world networks. Here we propose a local method for detecting communities in networks. The method is described as 'local' because it is intended to find the community to which a given source node belongs without knowing all the communities in the network. We have devised this method inspired by possible mechanisms for stable propagation of neuronal activities in neural networks. To demonstrate the effectiveness of our method, local detection of communities in synthetic benchmark networks and real social networks is examined. The community structure detected by our method is perfectly consistent with the correct community structure of these networks.

Keywords: Complex network, Community detection, Markov chain, Spreading activation, Neural network.

1 Introduction

In literature of network science, 'community' refers to a group of nodes that are densely connected within this group but are sparsely connected with nodes outside this group. Community structure is a hallmark of a variety of social, biological and engineering networks. Development of algorithms to detect communities in networks has been a focus of network science in the last decade [1, 2].

Many algorithms for community detection have already been proposed. Nevertheless, most of them are intended to exhaustively detect all the communities in the network. This type of community detection is computationally infeasible especially for large and dynamically evolving networks such as the World Wide Web or Facebook networks.

Another strategy of community detection, for which entire structure of the network is unnecessary to know, has drawn much interest recently. Algorithms based on this strategy are intended to find the community to which a source node belongs [3-8]. Starting from a given source node, one explores the network crawling links; exploration is continued until certain criteria are satisfied; the explored region, or a part of it, is then judged as the community to which the source node belongs. This

© Springer-Verlag Berlin Heidelberg 2013

type of community detection is described as ‘local’ because it requires knowledge only about the explored region of the network.

In this study we propose a novel method for local detection of communities. This method makes use of the dynamics that models stable propagation of neuronal activities in neural networks. The effectiveness of this method is examined using synthetic benchmark networks and real social networks.

2 Methods

2.1 Neural-Network Dynamics

Let $\mathbf{A} = (A_{nm})$ $(n, m = 1, \cdots, N)$ be the adjacency matrix of a network from which we wish to detect communities. Here, N is the number of nodes comprising the network. If nodes n and m are connected, $A_{nm} = A_{mn} = 1$; otherwise $A_{nm} = A_{mn} = 0$. In the present study we consider only networks having undirected links.

Now we define the ‘potential’ and the ‘activity’ for each node. Let $p_n(t)$ be the potential of node n at time t. Then the activity is given by

$$f(p_n(t)) = \frac{p_n(t)}{1 + \exp\left(-\beta\left(p_n(t) - \theta\right)\right)} . \tag{1}$$

As x increases, $y = f(x)$ sigmoidally rises and then asymptotically approaches $y = x$ (Fig. 1). This relationship between x and $f(x)$ is well known as the relationship between the membrane potential and the mean firing rate of a neuron [9].

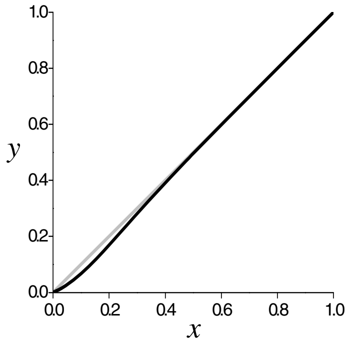


Fig. 1. As x increases, $y = f(x) = x/\left(1 + \exp\left(-\beta(x - \theta)\right)\right)$ (black line) sigmoidally rises and then asymptotically approaches $y = x$ (grey line). For this illustration we have chosen $\beta = 10$ and $\theta = 1/34$.

We suppose that activities propagate from nodes to nodes along links in an analogy to propagation of neuronal activities in networks of neurons. Additionally we postulate that throughout the propagation the sum, $\sum_{n=1}^{N} p_n(t)$, is kept constant with time. This condition will keep the network from falling into pathological states such as flare-up or burning-out of activities. Without loss of generality we set

$$\sum_{n=1}^{N} p_n(t) = 1 . \tag{2}$$

We assume that the propagation of activities is described by the equation

$$p_n(t) = \sum_{m=1}^{N} T_{nm} f\left(p_m(t-1)\right) + \frac{f\left(p_n(t-1)\right)}{\sum_{m=1}^{N} f\left(p_m(t-1)\right)}\left[1 - \sum_{m=1}^{N} f\left(p_m(t-1)\right)\right] \tag{3}$$

where $T_{nm} = A_{nm} \big/ \sum_{n'=1}^{N} A_{n'm}$. One can easily verify that condition (2) is derived from equation (3). The first term on the right-hand side of (3) means that the potential of node n is partly given by the sum of activities propagating from nodes that link to node n. If the right-hand side has this term only, $\sum_{n=1}^{N} p_n(t)$ would decrease from $\sum_{n=1}^{N} p_n(t-1)$ by $\sum_{n=1}^{N} p_n(t-1) - \sum_{n=1}^{N} f\left(p_n(t-1)\right)$. The second term is added to compensate for this decrease so that condition (2) is held; the decrement $1 - \sum_{m=1}^{N} f\left(p_m(t-1)\right)$ is redistributed to each node in such a way that the distribution to node n is proportional to its activity $f\left(p_n(t-1)\right)$. Thus, equation (3) can be viewed as phenomenological description of stable propagation of neuronal activities kept from falling into flare-up or burning-out.

It is instructive to give another interpretation to equation (3). In the limit $\theta \to 0$ and $\beta \to \infty$, equation (3) becomes

$$p_n(t) = \sum_{m=1}^{N} T_{nm} p_m(t-1) , \tag{4}$$

which is just the Markov-chain equation [10-12]. Hence one can suppose a ‘random walker’ who is wandering in the network; $p_n(t)$ is then interpreted as the probability that the random walker stays at node n at time t. For $\theta > 0$ and finite β, $p_n(t)$ can no longer be interpreted as the probability for a single random walker. Instead one can suppose many random walkers; $p_n(t)$ is now the population density of node n at time t. A random walker staying at node m at time $t-1$ monitors the population density of this node. With probability $r = 1 \big/ \left(1 + \exp\left(-\beta\left(p_m(t-1) - \theta\right)\right)\right)$ he/she selects one of the nodes linked from node m and then moves to the selected node at time t; with probability $1-r$ the random walker jumps to any node. For the latter, to which node he/she jumps is determined by the relative amplitude of the activity; namely, he/she jumps to node n with probability $f\left(p_n(t-1)\right) \big/ \sum_{m=1}^{N} f\left(p_m(t-1)\right)$.

2.2 Local Detection of Communities

The community to which a given source node (say, node l) belongs is detected by our algorithm as follows. First we set the initial condition

$$p_l(0)=1, \quad p_n(0)=0 \quad (n \neq l)\,. \tag{5}$$

As time passes activities, initially concentrated at the source node, spread in the network according to equation (3). Activities preferentially propagate within the community to be detected, where nodes are densely connected. Potentials of nodes at peripheral regions of the community are small because these nodes have less links than those centrally located in the community. Because of the sigmoidal rise of $f(x)$ at small x, activities of nodes at peripheral regions rapidly decay. Thus activities no more spread far beyond the peripheral regions. In the steady state of activity propagation, therefore, activities are localized within the community.

Iterative calculation of equation (3) with initial condition (5) eventually leads to the steady-state distribution of potentials, $\vec{p}^{\,(\mathrm{stead})}=\left\{p_1^{(\mathrm{stead})},\cdots,p_N^{(\mathrm{stead})}\right\}$. This steady-state distribution represents the community to which the source node belongs. The $p_n^{(\mathrm{stead})}$ has a graded value ranging from 0 to 1, which expresses the rank or the level of relative importance of node n in the detected community.

If the link density is almost uniform everywhere in the explored portion of the network, the potential of each node would be $\sim 1/N_e$ where N_e is the number of nodes in the explored portion. If the network has community structure, the potential of a node belonging to the community within which activities are localized will be $>1/N_e$. We can therefore appropriately set $\theta=1/N_e$. Since the size of networks examined in this paper is not so large ($N=200$ or 34), we set $N_e=N$.

Our algorithm detects a community as an attractor of neural-network dynamic. Arenas et al. [13, 14] have also proposed to detect communities in networks by coupled-oscillator dynamics. In their approach community structure emerges as temporally evolving synchronization patterns, whereas a community locally detected by our algorithm is represented by a static pattern.

3 Results

3.1 Local Detection of Communities from a Synthetic Benchmark Network

First we evaluate the effectiveness of local detection of communities by our algorithm using synthetic benchmark networks. Lancichinetti et al. have developed a method for synthesizing networks that have community structure [15, 16]. The number of communities and their sizes can be controlled by adjusting the parameter values. Here we have synthesized a network of $N=200$ nodes that has six communities. The parameter values used for synthesizing this network and the statistics of the communities are given in Appendix.

For each of the 200 nodes we have detected the community using our algorithm. For each node taken as a source, the steady state distribution falls into any of K

patterns, with K being the number of patterns for detected communities. We assume that each pattern corresponds to the community to which the source node belongs.

The number of communities K depends on β (Fig. 2). For a wide range of β the correct number (six) of communities is obtained. The detected community for each node taken as a source and the correct community to which this node belongs are compared in Fig. 3, which shows that local detection of communities in the synthesized benchmark network by our algorithm is perfect.

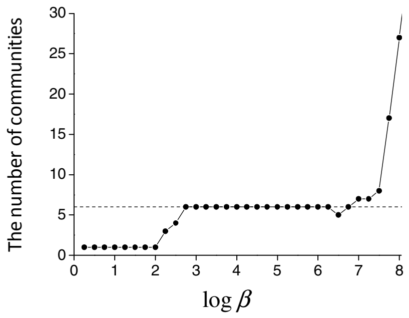


Fig. 2. The number of communities found in the synthesized benchmark network by our algorithm depends on β. Note that for a wide range of β the correct number of communities (six, indicated by broken line) is stably obtained.

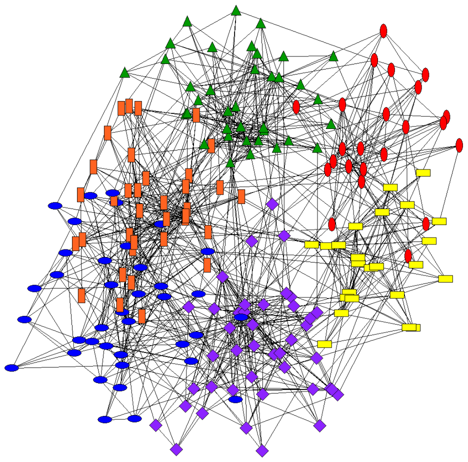


Fig. 3. Local detection of communities in the synthetic benchmark network. The color (red, blue, green, yellow, orange or purple) of each icon indicates the community detected by our algorithm for this node taken as a source. The shape (vertically long ellipse, horizontally long ellipse, vertically long box, horizontally long box, triangle or diamond) of each node indicates the correct community to which this node belongs. Note that nodes that have the same shape also have the same color. Here, we have chosen $\beta = e^5$.

3.2 Local Community Detection from the Zachary Karate-Club Network

The Zachary karate-club network [17] is a famous benchmark network that has been used by researchers developing algorithms for detection of social sub-groups. This is a real social network of $N = 34$ nodes with each representing a member of a karate club at a US university observed by Zachary. Each link indicates social interaction between two members. During the period of observation by Zachary, a dispute between the head teacher and the administrator had developed, which resulted in factional separation of the club into two groups with one led by the head teacher and the other led by the administrator. The goal of our task is to predict to which group each member belongs after the breakdown of the club.

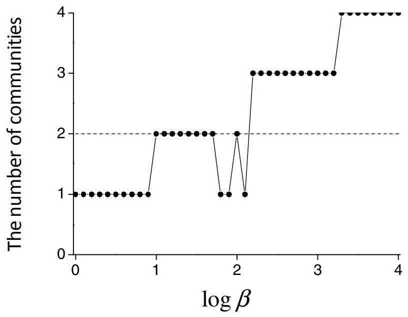


Fig. 4. The number of communities detected by our algorithm depends on β. Note that for a certain range of β the correct number of communities (two, indicated by broken line) is obtained.

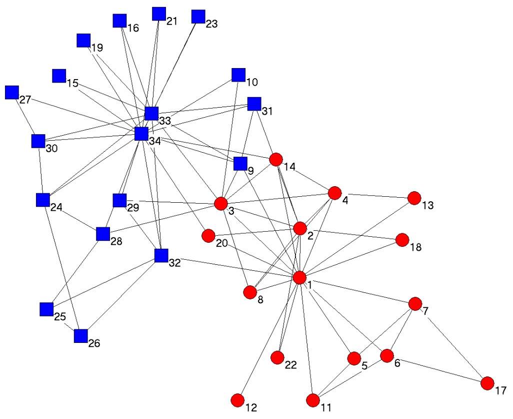


Fig. 5. Local detection of community structure in the Zachary karate-club network. The color (red or blue) of each icon indicates the community detected by our method for this node taken as a source. The shape (circle or box) of each node indicates the correct group to which the member corresponding to this node belongs. Note that nodes that have the same shape also have the same color. Here, we have chosen $\beta = e^{1.5}$.

The number of communities detected by our algorithm depends on β (Fig.4). For a wide range of β the correct number (two) of communities is obtained. The detected community for each node taken as a source and the correct group to which the member corresponding to this node belongs are compared in Fig. 5, which indicates that our algorithm perfectly predicts the groups to which individual members belong after factional separation of the club.

4 Discussion

We have proposed a novel method for uncovering local community structure in networks, which is inspired by possible mechanisms of stable propagation of neuronal activity in networks of neurons. We have shown that the local communities in the synthesized benchmark network can be detected correctly by our algorithm (Fig. 3). Application of this algorithm to a real social network, the Zachary karate club network, has perfectly replicated the factional separation of this club (Fig. 5). These results demonstrate the effectiveness of local detection of communities by our method.

The computational cost of local detection of communities by previously proposed algorithms is $\sim O\left(N_e^{\,3}\right)$ or $\sim O\left(N_e^{\,2} < k >\right)$ [3, 4], where N_e is the number of nodes in the explored portion of the network and $< k >$ is the mean number of links attached to individual nodes. On the other hand the cost of computation by our algorithm basically scales with the number of links within the explored region. This suggests that our algorithm is computationally more efficient than previously proposed ones, which will be more quantitatively examined in the forthcoming study.

The algorithm has parameters β and θ, and the number of detected communities depends on the values of these parameters (Fig. 2 and Fig. 4). In this study we have determined these values in a heuristic way. Introduction of some kind of measure to estimate the goodness of community detection, such as the 'modularity' [4, 18], might be useful for developing more principled ways of determining the values of these parameters.

Networks dealt with in the present study are restricted to those having undirected links. Local detection of communities in networks having directed links is an important issue to be addressed in the next step of our study. For this, introduction of random jump from sink nodes to any other nodes, the prescription used for calculation of the PageRank values in the World Wide Web [19], might be useful.

Acknowledgments. This study was partly supported by KAKENHI (23500379) and KAKENHI (23300061).

References

1. Fortunato, S.: Community detection in graphs. Phys. Rep. 486, 75–174 (2010)
2. Newman, M.E.J.: Communities, modules and large-scale structure in networks. Nature Phys. 8, 25–31 (2012)

3. Bagrow, J.P., Bollt, E.M.: Local method for detecting communities. Phys. Rev. E 72, 046108 (2005)
4. Clauset, A.: Finding local community structure in networks. Phys. Rev. E 72, 026132 (2005)
5. Luo, F., Wang, J., Promislow, E.: Exploring local community structures in large networks. Web Intelligence and Agent Systems 6, 387–400 (2008)
6. Chen, J., Zaïane, O., Goebel, R.: Local community identification in social networks. In: International Conference on Advances in Social Network Analysis and Mining (ASONAM 2009), pp. 237–242 (2009)
7. Lancichinetti, A., Fortunato, S., Kertesz, J.: Detecting the overlapping and hierarchical community structure in complex networks. New J. Phys. 11, 033015 (2009)
8. Chen, Q., Wu, T.-T., Fang, M.: Detecting local community structures in complex networks based on local degree central nodes. Physica A 392, 529–537 (2013)
9. Tuckwell, H.: Introduction to theoretical neurobiology: vol. 2 nonlinear and stochastic theories. Cambridge University Press (1988)
10. Collins, A.M., Loftus, E.F.: Spreading-Activation Theory of Semantic Processing. Psychological Review 82, 407–428 (1975)
11. Okamoto, H.: Topic-dependent document ranking: citation network analysis by analogy to memory retrieval in the brain. In: Honkela, T. (ed.) ICANN 2011, Part I. LNCS, vol. 6791, pp. 371–378. Springer, Heidelberg (2011)
12. Branting, L.K.: Context-sensitive detection of local community structure. Soc. Netw. Anal. Min. 2, 279–289 (2012)
13. Arenas, A., Diaz-Guilera, A., Perez-Vicente, C.J.: Synchronization reveals topological scales in complex networks. Phys. Rev. Lett. 96, 114102 (2006)
14. Arenas, A., Diaz-Guilera, A., Perez-Vicente, C.J.: Synchroization processes in complex networks. Physica D 224, 27–34 (2006)
15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms Phys. Rev. E 78, 046110 (2008)
16. `http://santo.fortunato.googlepages.com/benchmark.tgz`
17. Zachary, W.W.: An information flow model for conflict and fission in small groups. J. Anthropol. Res. 33, 291–473 (1977)
18. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113 (2004)
19. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Library Technologies Project (1998), `http://google.stanford.edu/~backrub/pageranksub.ps`

Appendix: Synthetic Benchmark Network

The benchmark network used in **3.1** was synthesized using the software downloaded from [16] under the following settings: Number of nodes 200; average degree 10; maximum degree 30; exponent for the degree distribution 2; exponent for the community size distribution 1; mixing parameter 0.2; minimum for the community sizes 20; maximum for the community sizes 50. The synthesized network has six communities having 23, 24, 36, 37, 38 and 42 nodes.

The Super-Turing Computational Power of Interactive Evolving Recurrent Neural Networks

Jérémie Cabessa and Alessandro E.P. Villa

Department of Information Systems
University of Lausanne
CH-1015 Lausanne, Switzerland

Abstract. Understanding the dynamical and computational capabilities of neural models represents an issue of central importance. Here, we consider a model of first-order recurrent neural networks provided with the possibility to evolve over time and involved in a basic interactive and memory active computational paradigm. In this context, we prove that the so-called *interactive evolving recurrent neural networks* are computationally equivalent to interactive Turing machines with advice, hence capable of super-Turing potentialities. We further provide a precise characterisation of the ω-translations realised by these networks. Therefore, the consideration of evolving capabilities in a first-order neural model provides the potentiality to break the Turing barrier.

Keywords: recurrent neural networks, neural computation, interactive computation, analog computation, Turing machines with advice, super-Turing.

1 Introduction

Unerstanding the dynamical and computational capabilities of neural models represents an issue of central importance to assess the performances at reach by neural networks. In this context, much interest has been focused on comparing the computational capabilities of diverse theoretical neural models to those of abstract computing devices [7,14,6,9,8,11,12,10]. As a consequence, the computational power of neural networks has been shown to be intimately related to the nature of their synaptic weights and activation functions, hence capable to range from finite state automata up to super-Turing capabilities.

However, in this global line of thinking, the neural models which have been considered fail to capture some essential biological features that are significantly involved in the processing of information in the brain. In particular, the plasticity of biological neural networks as well as the interactive nature of information processing in bio-inspired complex systems have only recently started to be investigated [2,3].

The present paper falls within this perspective and concerns the computational capabilities of a model of interactive evolving recurrent neural networks. This work is a direct extension of previous results by Cabessa [1]. More precisely,

© Springer-Verlag Berlin Heidelberg 2013

we consider a model of evolving recurrent neural networks where the synaptic strengths of the neurons can change over time rather than staying static, and we study the computational capabilities of such networks in a basic context of interactive computation, in line with the framework proposed by van Leeuwen and Wiedermann [15,17]. In this context, we prove that rational- and real-weighted *interactive evolving recurrent neural networks* are both computationally equivalent to interactive Turing machines with advice, hence capable of super-Turing capabilities. Moreover, we provide a precise mathematical characterisation of the ω-translations realised by these neural models. These results support the idea that some intrinsic feature of biological intelligence might be beyond the scope of the current state of artificial intelligence, and that the concept of evolution might be strongly involved in the computational capabilities of biological neural networks. They also show that the nature of the synaptic weights has no influence on the computational power of interactive evolving neural networks.

2 Preliminaries

Given some finite alphabet Σ, we let Σ^*, Σ^+, Σ^n, and Σ^ω denote respectively the sets of finite words, non-empty finite words, finite words of length n, and infinite words, all of them over alphabet Σ. We also let $\Sigma^{\leq\omega} = \Sigma^* \cup \Sigma^\omega$ be the set of all possible words (finite or infinite) over Σ. The empty word is denoted by λ.

For any $x \in \Sigma^{\leq\omega}$, the *length* of x is denoted by $|x|$ and corresponds to the number of letters contained in x. If x is non-empty, we let $x(i)$ denote the $(i+1)$-th letter of x, for any $0 \leq i < |x|$. The *prefix* $x(0)\cdots x(i)$ of x is denoted by $x[0{:}i]$, for any $0 \leq i < |x|$. For any $x \in \Sigma^*$ and $y \in \Sigma^{\leq\omega}$, the fact that x is a *prefix* (resp. *strict prefix*) of y is denoted by $x \subseteq y$ (resp. $x \subsetneq y$). If $x \subseteq y$, we let $y - x = y(|x|)\cdots y(|y|-1)$ be the *suffix* of y that is not common to x (we have $y - x = \lambda$ if $x = y$). Moreover, the *concatenation* of x and y is denoted by $x \cdot y$ or sometimes simply by xy. The word x^n consists of n copies of x concatenated together, with the convention that $x^0 = \lambda$.

A function $f : \Sigma^* \to \Sigma^*$ is called *monotone* if the relation $x \subseteq y$ implies $f(x) \subseteq f(y)$, for all $x, y \in \Sigma^*$. It is called *recursive* if it can be computed by some Turing machine over Σ. Furthermore, throughout this paper, any function $\varphi : \Sigma^\omega \to \Sigma^{\leq\omega}$ will be referred to as an *ω-translation*.

Note that any monotone function $f : \Sigma^* \to \Sigma^*$ *induces in the limit* an ω-translation $f_\omega : \Sigma^\omega \to \Sigma^{\leq\omega}$ defined by

$$f_\omega(x) = \lim_{i\geq 0} f(x[0{:}i])$$

where $\lim_{i\geq 0} f(x[0{:}i])$ denotes the smallest finite word that contains each word of $\{f(x[0{:}i]) : i \geq 0\}$ as a finite prefix if $\lim_{i\to\infty} |f(x[0{:}i])| < \infty$, and $\lim_{i\geq 0} f(x[0{:}i])$ denotes the unique infinite word that contains each word of $\{f(x[0{:}i]) : i \geq 0\}$ as a finite prefix if $\lim_{i\to\infty} |f(x[0{:}i])| = \infty$. Note that the monotonicity of f ensures that the value $f_\omega(x)$ is well-defined for all $x \in \Sigma^\omega$. Intuitively, the value $f_\omega(x)$ corresponds to the finite or infinite word that is ultimately approached by the sequence of growing prefixes $\langle f(x[0{:}i]) : i \geq 0 \rangle$.

An ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ will be called *continuous* if there exists a monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$; it will be called *recursive continuous* if there exists a monotone and recursive function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$.

3 Interactive Computation

3.1 The Interactive Paradigm

Interactive computation refers to the computational framework where systems may react or interact with each other as well as with their environment during the computation [5]. This paradigm was theorised in contrast to classical computation [13] which rather proceeds in a closed-box fashion and was argued to "no longer fully corresponds to the current notions of computing in modern systems" [17]. Interactive computation also provides a particularly appropriate framework for the consideration of natural and bio-inspired complex information processing systems [15,17].

The general interactive computational paradigm consists of a step by step exchange of information between a system and its environment. In order to capture the unpredictability of next inputs at any time step, the dynamically generated input streams need to be modeled by potentially infinite sequences of symbols (the case of finite sequences of symbols would necessarily reduce to the classical computational framework) [18,17].

Throughout this paper, we consider a basic interactive computational scenario where at every time step, the environment sends a non-empty input bit to the system (full environment activity condition), the system next updates its current state accordingly, and then either produces a corresponding output bit, or remains silent for a while to express the need of some internal computational phase before outputting a new bit, or remains silent forever to express the fact that it has died. Consequently, after infinitely many time steps, the system will have received an infinite sequence of consecutive input bits and translated it into a corresponding finite or infinite sequence of not necessarily consecutive output bits. Accordingly, any interactive system $\mathcal{S}$ realises an ω-translation $\varphi_\mathcal{S} : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$.

3.2 Interactive Turing Machines

An *interactive Turing machine* (I-TM) $\mathcal{M}$ consists of a classical Turing machine yet provided with input and output ports rather than tapes in order to process the interactive sequential exchange of information between the device and its environment [15]. According to our interactive scenario, it is assumed that at every time step, the environment sends a non-silent input bit to the machine and the machine answers by either producing a corresponding output bit or rather remaining silent (expressed by the fact of outputting the λ symbol).

An *interactive Turing machine with advice* (I-TM/A) $\mathcal{M}$ consists of an interactive Turing machine provided with an advice mechanism which comes in the form of an *advice function* $\alpha : \mathbb{N} \to \{0,1\}^*$ [15]. Moreover, the machine $\mathcal{M}$ uses two auxiliary special tapes, an *advice input tape* and an *advice output tape*, as well as a designated *advice state*. During its computation, $\mathcal{M}$ can write the binary representation of an integer m on its advice input tape, one bit at a time. Yet at time step n, the number m is not allowed to exceed n. Then, at any chosen time, the machine can enter its designated advice state and then have the finite string $\alpha(m)$ be written on the advice output tape in one time step, replacing the previous content of the tape. The machine can repeat this extra-recursive calling process as many times as it wants during its infinite computation.

According to this definition, for any infinite *input stream* $s \in \{0,1\}^\omega$, we define the corresponding *output stream* $o_s \in \{0,1\}^{\leq\omega}$ of $\mathcal{M}$ as the finite or infinite subsequence of (non-λ) output bits produced by $\mathcal{M}$ after having processed input s. In this manner, any machine $\mathcal{M}$ naturally induces an ω-translation $\varphi_\mathcal{M} : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by $\varphi_\mathcal{M}(s) = o_s$, for each $s \in \{0,1\}^\omega$. Finally, an ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ is said to be *realisable* by some interactive Turing machine with advice iff there exists some I-TM/A $\mathcal{M}$ such that $\varphi_\mathcal{M} = \psi$.

Interactive Turing machines with advice are strictly more powerful than interactive Turing machines (without advice) [15], and were shown to be computationally equivalent to several others other non-uniform models of interactive computation, like sequences of interactive finite automata, site machines, and web Turing machines [15].

4 Interactive Evolving Recurrent Neural Networks

An *evolving recurrent neural network* (Ev-RNN) consists of a synchronous network of neurons (or processors) related together in a general architecture. The network contains a finite number of neurons $(x_i)_{i=1}^N$, M parallel input lines $(u_i)_{i=1}^M$, and P designated output neurons among the N. Furthermore, the synaptic connections between the neurons are assumed to be time dependent rather than static. At each time step, the activation value of every neuron is updated by applying a linear-sigmoid function to some weighted affine combination of values of other neurons or inputs at previous time step.

Formally, given the activation values of the internal and input neurons $(x_j)_{j=1}^N$ and $(u_j)_{j=1}^M$ at time t, the activation value of each neuron x_i at time $t+1$ is then updated by the following equation

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t) \cdot u_j(t) + c_i(t)\right) \tag{1}$$

for $i = 1, \ldots, N$, where all $a_{ij}(t)$, $b_{ij}(t)$, and $c_i(t)$ are *time dependent* values describing the evolving weighted synaptic connections and weighted bias of the network, and σ is the classical saturated-linear activation function defined by $\sigma(x) = 0$ if $x < 0$, $\sigma(x) = x$ if $0 \leq x \leq 1$, and $\sigma(x) = 1$ if $x > 1$.

In order to stay consistent with our interactive scenario, we need to define the notion of an *interactive evolving recurrent neural network* (I-Ev-RNN) which adheres to a rigid encoding of the way input and output are interactively processed between the environment and the network.

First of all, we assume that any I-Ev-RNN is provided with a single binary input line u whose role is to transmit to the network the infinite input stream of bits sent by the environment. We also suppose that any I-Ev-RNN is equipped with two binary output lines, a data line y_d and a validation line y_v. The role of the data line is to carry the output stream of the network, while the role of the validation line is to describe when the data line is active and when it is silent. Accordingly, the output stream transmitted by the network to the environment will be defined as the (finite or infinite) subsequence of successive data bits that occur simultaneously with positive validation bits.

Hence, if $\mathcal{N}$ is an I-Ev-RNN with initial activation values $x_i(0) = 0$ for $i = 1, \ldots, N$, then any infinite *input stream* $s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$ transmitted to input line u induces via Equation (1) a corresponding pair of infinite streams $(y_d(0)y_d(1)y_d(2)\cdots, y_v(0)y_v(1)y_v(2)\cdots) \in \{0,1\}^\omega \times \{0,1\}^\omega$. The *output stream* of $\mathcal{N}$ according to input s is then given by the finite or infinite subsequence o_s of successive data bits that occur simultaneously with positive validation bits, namely $o_s = \langle y_d(i) : i \in \mathbb{N} \text{ and } y_v(i) = 1\rangle \in \{0,1\}^{\leq\omega}$. It follows that any I-Ev-RNN $\mathcal{N}$ naturally induces an ω-translation $\varphi_\mathcal{N} : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by $\varphi_\mathcal{N}(s) = o_s$, for each $s \in \{0,1\}^\omega$. An ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ is said to be *realisable* by some I-Ev-RNN iff there exists some I-Ev-RNN $\mathcal{N}$ such that $\varphi_\mathcal{N} = \psi$.

Finally, throughout this paper, two models of interactive evolving recurrent neural networks are considered according to whether their underlying synaptic weights are confined to the class of rational or real numbers. Rational- and real-weighted interactive evolving recurrent neural network will be dented by I-Ev-RNN[$\mathbb{Q}$] and I-Ev-RNN[$\mathbb{R}$], respectively. Note that since rational numbers are included in real numbers, every I-Ev-RNN[$\mathbb{Q}$] is also a particular I-Ev-RNN[$\mathbb{R}$] by definition.

5 The Computational Power of Interactive Evolving Recurrent Neural Networks

The following result states that interactive evolving recurrent neural networks are computationally equivalent to interactive Turing machine with advice, irrespective of whether their synaptic weights are rational or real. A precise mathematical characterisation of the ω-translations realised by these networks is also provided. It directly follows that interactive evolving neural networks are capable super-Turing computational potentialities.

Theorem 1. *Let $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ be an ω-translation. The following conditions are equivalent:*

1. ψ is realisable by some I-TM/A;
2. ψ is realisable by some I-Ev-RNN[$\mathbb{Q}$];
3. ψ is realisable by some I-Ev-RNN[$\mathbb{R}$];
4. ψ is continuous.

Proof (sketch). (1) $\Rightarrow$ (2): We will use the fact that every Turing machine can be simulated by some rational-weighted recurrent neural network [12]. Let $\mathcal{M}$ be some I-TM/A with advice function $\alpha : \mathbb{N} \to \{0,1\}^*$. We show that there exists an I-Ev-RNN[$\mathbb{Q}$] $\mathcal{N}$ which realises the same ω-translation as $\mathcal{M}$. First, for each $i > 0$, let q_i be a rational number encoding in a recursive manner the sequence of successive advice values $\langle \alpha(0), \dots, \alpha(i) \rangle$. Note that such an encoding is indeed possible since by definition of α every $\alpha(i)$ is a finite word. Now, consider the following description of an I-Ev-RNN[$\mathbb{Q}$] $\mathcal{N}$. The network $\mathcal{N}$ contains a specific evolving synaptic connection which takes as evolving weights the successive values q_i's defined above, for all $i > 0$. The network $\mathcal{N}$ also contains a non-evolving rational-weighted part which is designed is order to simulate $\mathcal{M}$ as follows: every recursive computational step of $\mathcal{M}$ is simulated by $\mathcal{N}$ in the classical way, as described in [12]; moreover, for every extra-recursive call to some advice value $\alpha(m)$ performed by $\mathcal{M}$ at some time $t \geq m$, $\mathcal{N}$ first waits for the synaptic weight q_t to occur, then stores the synaptic weight q_t in its memory, then decodes the specific string $\alpha(m)$ from the rational value q_t (which is possible since $t \geq m$), and then pursues the simulation of the next recursive steps of $\mathcal{M}$ in the classical way [12]. In this manner, $\mathcal{N}$ realises the same ω-translation as $\mathcal{M}$.

(2) $\Rightarrow$ (3): Note that every I-Ev-RNN[$\mathbb{Q}$] is also an I-Ev-RNN[$\mathbb{R}$] by definition. Hence, if ψ is an ω-translation realised by some I-Ev-RNN[$\mathbb{Q}$] $\mathcal{N}$, then ψ is also realised by some I-Ev-RNN[$\mathbb{R}$], namely by $\mathcal{N}$ itself.

(3) $\Rightarrow$ (4): Let $\varphi_{\mathcal{N}}$ be an ω-translation realised by some I-Ev-RNN[$\mathbb{R}$] $\mathcal{N}$. We show that $\varphi_{\mathcal{N}}$ is continuous. For this purpose, consider the function $f : \{0,1\}^* \to \{0,1\}^*$ which maps every finite word u to the unique corresponding finite word output by $\mathcal{N}$ after precisely $|u|$ steps of computation, when $u \cdot x$ is provided as input bit by bit, for any possible suffix $x \in \{0,1\}^\omega$. The definition of our interactive scenario ensures that f is well-defined (i.e., that $f(u)$ is independent of the suffix x), and that f is monotone. We can prove that the function $\varphi_{\mathcal{N}}$ realised by $\mathcal{N}$ corresponds precisely to the limit of the monotone function f as defined in Section 2, or in other words, that $\varphi_{\mathcal{N}} = f_\omega$. Therefore, $\varphi_{\mathcal{N}}$ is continuous.

(4) $\Rightarrow$ (1): Let $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ be some continuous ω-translation. Then, by definition, there exists some monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$. For each $i \geq 0$, let $(z_{i,j})_{j=1}^{2^i}$ be the lexicographic enumeration of all binary words of length i. Let $\alpha : \mathbb{N} \to \{0,1\}^*$ be the advice function such that $\alpha(i)$ represents some recursive encoding of the successive values $f(z_{i,j})$ separated by $\sharp$'s, for $j = 1, \dots, 2^i$ (for instance, $\alpha(2)$ is a binary encoding of $\sharp f(00) \sharp f(01) \sharp f(10) \sharp f(11) \sharp$). Now, consider the I-TM/A $\mathcal{M}$ with advice α working on every infinite input $s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$ as follows: for each new input bit $s(i+1)$, $\mathcal{M}$ calls its advice value $\alpha(i+1)$, decodes the specific value $f(s(0)\cdots s(i+1))$ from it, checks if $f(s(0)\cdots s(i+1))$ strictly extends the

previous decoded value $f(s(0)\cdots s(i))$, and if this is the case, outputs this extension bit by bit. We can show that the function $\varphi_{\mathcal{M}}$ realised by $\mathcal{M}$ in this manner corresponds precisely to the limit of the monotone function f as defined in Section 2, or in other words, that $\varphi_{\mathcal{M}} = f_\omega$. Yet since $f_\omega = \psi$, one has $\varphi_{\mathcal{M}} = \psi$, meaning that ψ is realised by the I-TM/A $\mathcal{M}$. □

6 Discussion

The present paper provides a complete mathematical characterisation of the computational power of evolving recurrent neural networks involved in a basic context of interactive and memory active computation. It is shown that interactive evolving neural networks are computationally equivalent to interactive machines with advice, hence capable of super-Turing potentialities, irrespective of whether their underlying synaptic weights are rational or real.

These results show that the consideration of evolving capabilities in a first-order neural model provides the potentiality to break the Turing barrier. The super-Turing computational equivalence between I-Ev-RNN[$\mathbb{Q}$]s and I-Ev-RNN[$\mathbb{R}$]s reveals two important considerations. First, the incorporation of the power of the continuum in the model does not increase further the computational capabilities of the networks. This feature supports the extension of the Church-Turing Thesis to the context of interactive computation stated by van Leeuwen and Wiedermann [16]:

> "Any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice."

Second and most importantly, the super-Turing computational capabilities can be achieved without the need of a framework based on the power of the continuum – in the case of the Ev-RNN[$\mathbb{Q}$] model. This feature is particularly meaningful, since while the power of the continuum is a pure conceptualisation of the mind, the evolving capabilities of the networks are, by contrast, really observable in nature. However, note that such super-Turing capabilities can only be achieved in cases where the evolving synaptic patters are themselves non-recursive (i.e., non Turing-computable). The question of the existence in nature of such non-recursive patterns of evolution remains beyond the scope of this paper. We refer to Copeland's extensive work for deeper philosophical considerations about hypercomputation in general [4].

From a general perspective, we believe that such theoretical studies about the computational power of bio-inspired neural models might ultimately bring further insight to the understanding of the intrinsic natures of both biological as well as artificial intelligences. We also think that foundational approaches to alternative models of computation might in the long term not only lead to relevant theoretical considerations, but also to important practical applications. Similarly to the theoretical work from Turing which played a crucial role in the practical realisation of modern computers, further foundational considerations of alternative models of computation will certainly contribute to the emergence of novel computational technologies and computers, and step by step, open the way to the next computational era.

References

1. Cabessa, J.: Interactive evolving recurrent neural networks are super-turing. In: Filipe, J., Fred, A.L.N. (eds.) ICAART (1), pp. 328–333. SciTePress (2012)
2. Cabessa, J., Siegelmann, H.T.: Evolving recurrent neural networks are super-turing. In: IJCNN, pp. 3200–3206. IEEE (2011)
3. Cabessa, J., Siegelmann, H.T.: The computational power of interactive recurrent neural networks. Neural Computation 24(4), 996–1019 (2012)
4. Jack Copeland, B.: Hypercomputation. Minds Mach. 12(4), 461–502 (2002)
5. Goldin, D., Smolka, S.A., Wegner, P.: Interactive Computation: The New Paradigm. Springer-Verlag New York, Inc., Secaucus (2006)
6. Kleene, S.C.: Representation of events in nerve nets and finite automata. In: Shannon, C., McCarthy, J. (eds.) Automata Studies, pp. 3–41. Princeton University Press, Princeton (1956)
7. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysic 5, 115–133 (1943)
8. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Inc., Englewood Cliffs (1967)
9. von Neumann, J.: The computer and the brain. Yale University Press, New Haven (1958)
10. Siegelmann, H.T.: Neural networks and analog computation: beyond the Turing limit. Birkhauser Boston Inc., Cambridge (1999)
11. Siegelmann, H.T., Sontag, E.D.: Analog computation via neural networks. Theor. Comput. Sci. 131(2), 331–360 (1994)
12. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. J. Comput. Syst. Sci. 50(1), 132–150 (1995)
13. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc. 2(42), 230–265 (1936)
14. Turing, A.M.: Intelligent machinery. Technical report, National Physical Laboratory, Teddington, UK (1948)
15. van Leeuwen, J., Wiedermann, J.: Beyond the Turing Limit: Evolving Interactive Systems. In: Pacholski, L., Ružička, P. (eds.) SOFSEM 2001. LNCS, vol. 2234, pp. 90–109. Springer, Heidelberg (2001)
16. van Leeuwen, J., Wiedermann, J.: The Turing machine paradigm in contemporary computing. In: Engquist, B., Schmid, W. (eds.) Mathematics Unlimited - 2001 and Beyond. LNCS, pp. 1139–1155. Springer, Heidelberg (2001)
17. Wiedermann, J., van Leeuwen, J.: How we think of computing today. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 579–593. Springer, Heidelberg (2008)
18. Wegner, P.: Interactive foundations of computing. Theor. Comput. Sci. 192, 315–351 (1998)

Group Fused Lasso

Carlos M. Alaíz, Álvaro Barbero, and José R. Dorronsoro

Dpto. Ingeniería Informática & Inst. Ingeniería del Conocimiento,
Universidad Autónoma de Madrid, 28049 Madrid, Spain
{carlos.alaiz,alvaro.barbero,jose.dorronsoro}@uam.es

Abstract. We introduce the Group Total Variation (GTV) regularizer, a modification of Total Variation that uses the $\ell_{2,1}$ norm instead of the ℓ_1 one to deal with multidimensional features. When used as the only regularizer, GTV can be applied jointly with iterative convex optimization algorithms such as FISTA. This requires to compute its proximal operator which we derive using a dual formulation. GTV can also be combined with a Group Lasso (GL) regularizer, leading to what we call Group Fused Lasso (GFL) whose proximal operator can now be computed combining the GTV and GL proximals through Dykstra algorithm. We will illustrate how to apply GFL in strongly structured but ill-posed regression problems as well as the use of GTV to denoise colour images.

Keywords: Group Fused Lasso, Group Total Variation, Group Lasso, Fused Lasso, Total Variation.

1 Introduction

The irruption of big data, i.e., the need to study problems having very large sample sizes or very large dimensions or both, has resulted in a renewed interest in linear models, either because processing large samples with non-linear models is computationally demanding, or because a large dimension yields rich enough patterns so that methods enlarging pattern dimension such as the kernel trick add marginal value. Among linear models, Mean Square Error is the simplest fitting function, although it is well known that some regularizer has to be added, either to ensure good generalization, or just because the initial problem may be ill-posed. Classic choices include $\|w\|_2^2$ (ridge regression) and $\|w\|_1$ (Lasso [8]), and recently more ℓ_1-based regularizers such as Group Lasso [10] or Fused Lasso [9], have been introduced.

From a general point of view all these models can be stated as the problem of finding a $w^* \in \mathbb{R}^M$ which minimizes a certain functional $f(w) = f_{\mathrm{L}}(w) + f_{\mathrm{R}}(w)$ of the weights, with f_{R} the regularization term which somehow bounds the complexity of the model and f_{L} the loss functional. In more detail, assume a training set composed by P input patterns, $\{x^p\}_{p=1}^P$, with $x^p \in \mathbb{R}^M$, and their corresponding targets $\{y^p\}_{p=1}^P$, $y^p \in \mathbb{R}$. If $X \in \mathbb{R}^{P\times M}$ is the matrix having input patterns as rows and $y \in \mathbb{R}^P$ is the target vector, the overall problem for square loss can be written as

© Springer-Verlag Berlin Heidelberg 2013

$$\min_{w\in\mathbb{R}^M} f(w) = \min_{w\in\mathbb{R}^M} f_{\mathrm{L}}(w) + \lambda f_{\mathrm{R}}(w) = \min_{w\in\mathbb{R}^M} \|Xw-y\|_2^2 + \lambda f_{\mathrm{R}}(w), \qquad (1)$$

where λ is a parameter to control the strength of the regularizer.

Taking $f_{\mathrm{R}}(w) = \|w\|_1 = \sum_{i=1}^M |w_i|$ results in the Lasso approach (LA), which enforces sparsity in the coefficients with an implicit feature selection, since only those inputs corresponding to nonzero coefficients have an impact in the model.

In some problems the features can present a spatial structure which we may want the models to capture. One way to do this is to enforce similarity among the coefficients corresponding to nearby features. If we do not consider any multidimensional feature structure, this can be achieved using a Total Variation (TV) regularizer $\mathrm{TV}_1(w) = \sum_{i=2}^M |w_i - w_{i-1}|$, which penalizes the differences between consecutive coefficients. Some sparsity may also be wanted and the overall regularizer to be used is then $f_{\mathrm{R}}(w) = \|w\|_1 + \hat{\lambda}\,\mathrm{TV}_1(w) = \|w\|_1 + \hat{\lambda}\|Dw\|_1$, where $D \in \mathbb{R}^{(M-1)\times M}$ is the differencing matrix with $D_{i,i} = -1$, $D_{i,i+1} = 1$ and $D_{ij} = 0$ elsewhere. The resulting model is called the Fused Lasso (FL).

Neither LA nor FL do consider any possible group structure on the problem features and, therefore, the resulting models will not reflect it even if it may be present. Assume, however, that the pattern features x have such a group structure. We may then see x as a collection of multidimensional features, that is, x has NV components that come in N groups with V features each and therefore $x = (x_{1,1}, x_{1,2}, \ldots, x_{1,V}, x_{2,1}, x_{2,2}, \ldots x_{2,V}, \ldots, x_{N,1}, x_{N,2}, \ldots, x_{N,V})^\top \in \mathbb{R}^{NV}$. The first subscript in $x_{n,v}$ indicates the group (or the multidimensional feature) and the second subscript the group feature so x is decomposed in N blocks $x_n = (x_{n,1}, \ldots, x_{n,V})^\top$ that contain V variables. The mixed $\ell_{2,1}$ norm is possibly the easiest and most natural regularizer in this framework. More precisely, for a vector w with the above group structure, its $\ell_{2,1}$ norm $\|w\|_{2,1}$ is defined as $\|w\|_{2,1} = \sum_{n=1}^N \|w_n\|_2$, which is just the ℓ_1 norm of the ℓ_2 group norms. This leads to the Group Lasso model (GL) whose regularizer is then $f_{\mathrm{R}}(w) = \|w\|_{2,1}$.

In this work we will extend GL to a fused setting, introducing first a new Group Total Variation regularizer (GTV) defined as:

$$\mathrm{GTV}(w) = \sum_{n=2}^N \sqrt{\sum_{v=1}^V (w_{n,v} - w_{n-1,v})^2},$$

and considering a full regularization functional that adds the GTV term to the standard $\ell_{2,1}$ regularizer of GL. We can write it in compact notation as

$$f_{\mathrm{R}}(w) = \|w\|_{2,1} + \hat{\lambda}\|\bar{D}w\|_{2,1}, \text{ with } \bar{D} = \begin{pmatrix} -I & I & & \\ & \ddots & \ddots & \\ & & -I & I \end{pmatrix}. \qquad (2)$$

$\bar{D} \in \mathbb{R}^{(N-1)V\times NV}$ is the group differencing matrix, and $I \in \mathbb{R}^{V\times V}$ stands for the identity matrix. We call this model Group Fused Lasso (GFL). Notice that if $V = 1$ we recover FL, and if $V = M$, i.e., there is a single group with M

variables, GFL boils down to a variant of FL using a TV_2 regularizer, also known as ℓ_2-Variable Fusion [2].

We will solve the GFL optimization problem through convex proximal optimization techniques. We will essentially apply a variant of the FISTA algorithm which, in turn, requires that we can compute the proximal operator of the GFL regularizer, something we will do in Sect. 2. We point out that GFL with only the group $\|\bar{D}w\|_{2,1}$ penalty has been introduced in [6]. However, its solution is different from ours, as it reduces this GFL to a GL model that is then solved by a group LARS algorithm. We believe our approach to be better suited to deal with the full general GFL case. We shall illustrate the behaviour of GFL over two examples in Sect. 3, and we will close the paper in Sect. 4 with a discussion and pointers to further work.

2 Solving Group Fused Lasso with Proximal Methods

All the ℓ_1 regularizers of Sect. 1 lead to non-differentiable optimization problems, which prevents solving them by standard gradient-based methods. However, they fit very nicely under the paradigm of Proximal Methods (PMs) that we briefly review next. Recall that the function to be minimized in (1) is $f_{\mathrm{L}}(w) + f_{\mathrm{R}}(w)$, where we include the penalty factor λ in $f_{\mathrm{R}}(w)$.

Denote by $\partial h(w)$ the subdifferential at w of a convex function h; since both terms are convex and $f_{\mathrm{L}}(w)$ is differentiable, w^* will be a minimum of $f_{\mathrm{L}}(w) + f_{\mathrm{R}}(w)$ iff $0 \in \partial(f_{\mathrm{L}}(w^*) + f_{\mathrm{R}}(w^*))$ [3] or, by the Moreau–Rockafellar theorem, $0 \in \nabla f_{\mathrm{L}}(w^*) + \partial f_{\mathrm{R}}(w^*)$. Equivalently, we have $-\gamma \nabla f_{\mathrm{L}}(w^*) \in \gamma\lambda\partial f_{\mathrm{R}}(w^*)$ for any $\gamma > 0$ and, also, $w^* - \gamma\nabla f_{\mathrm{L}}(w^*) \in w^* + \gamma\partial f_{\mathrm{R}}(w^*) = (I + \gamma\partial f_{\mathrm{R}})(w^*)$. Thus, the set function $(I + \gamma\partial f_{\mathrm{R}})^{-1}$ verifies

$$w^* \in (I + \gamma\partial f_{\mathrm{R}})^{-1} (w^* - \gamma\nabla f_{\mathrm{L}}(w^*)) . \tag{3}$$

Now, if F is a convex, lower semicontinuous function, its proximal operator at w with step $\gamma > 0$ is defined as

$$z_w = \mathrm{prox}_{\gamma;F}(w) = \arg\min_{z\in\mathbb{R}^M} \left\{ \frac{1}{2}\|z - w\|_2^2 + \gamma F(z) \right\}.$$

Notice that then we have $0 \in z_w - w + \gamma\partial F(z_w)$, that is, $z_w \in (I + \partial F)^{-1}(w)$. For a general convex F, it can be shown [3] that ∂F is a monotone operator and, while in principle $(I+\partial F)^{-1}$ would be just a set-function, it is actually uniquely valued. Therefore, it defines a function for which $\mathrm{prox}_{\gamma;F}(w) = z_w = (I + \partial F)^{-1}(w)$ holds. Thus, going back to (3), it follows that $w^* = \mathrm{prox}_{\gamma;f_{\mathrm{R}}}(w^* - \gamma\nabla f_{\mathrm{L}}(w^*))$, which immediately suggests an iterative algorithm of the form

$$w^{k+1} = \mathrm{prox}_{\gamma;f_{\mathrm{R}}}\left(w^k - \gamma\nabla f_{\mathrm{L}}(w^k)\right) .$$

This is at the heart of the well known proximal gradient method [7] and of its ISTA and FISTA (Fast Iterative Shrinkage–Thresholding Algorithm) extensions [4]. In particular, we will focus on FISTA, based on the pair of equations:

$$w^k = \mathrm{prox}_{\frac{1}{K};f_{\mathrm{R}}}\left(z^k - \frac{1}{K}\nabla f_{\mathrm{L}}(z^k)\right), \quad z^{k+1} = w^k + \frac{t^k - 1}{t^{k+1}}(w^k - w^{k-1}),$$

where $t^{k+1} = \frac{1}{2}(1 + \sqrt{1 + 4t_k^2})$ and K is a the Lipschitz constant for ∇f_{L}. Notice that these algorithms require at each step the computation of the proximal operator at the current w^k. We discuss next these operators for GFL.

Observe that to solve problem (2) for the complete GFL regularizer, we need the proximal operator of the sum of the GTV and GL terms. Both regularizers are not separable, so their joint proximal operator cannot be built by the usual expedient of applying consecutively the proximal operators of GTV and GL. However, we can still solve the proximal problem by the Proximal Dykstra (PD) [7] algorithm, which allows to compute the proximal operator of the sum of several terms combining their individual proximal operators in an iterative fashion. Therefore we can focus on computing each proximal operator separately. In our case, the proximal operator of the GL regularizer is just the group soft-thresholding [1] defined as $\mathrm{prox}_{\gamma;\|\cdot\|_{2,1}}(w_{n,v}) = w_{n,v}(1 - \gamma/\|w_n\|_2)^+$, and we will derive now the proximal operator for GTV, following an analogous argument to the one in [2] for TV. We have to solve

$$\mathrm{prox}_{\gamma;\mathrm{GTV}}(w) = \mathrm{argmin}_{z \in \mathbb{R}^M} \frac{1}{2}\|z - w\|_2^2 + \gamma \|\bar{D}z\|_{2,1}, \tag{4}$$

which is a particular case of the more general problem $\inf_{z \in \mathbb{R}^M} \; f(z) + \gamma r(Bz)$, where $B \equiv \bar{D}$, $r(\cdot) \equiv \|\cdot\|_{2,1}$ and $f(y) \equiv \frac{1}{2}\|y - w\|_2^2$. In turn, this is equivalent to $\inf_{z,v} \; f(z) + \gamma r(v)$ s.t. $v = Bz$, with $z \in \mathbb{R}^M$ and $v \in \mathbb{R}^{(N-1)V}$. Writing its Lagrangian as $\mathcal{L}(z, v; u) = f(z) + \gamma r(v) + u \cdot (Bz - v)$ with $u \in \mathbb{R}^{(N-1)V}$, we can transform the equivalent saddle point problem $\inf_{z,v} \; \sup_u \; \mathcal{L}(z, v, u)$ into the dual problem

$$\inf_u \; f^*(-B^\top u) + \gamma r^* \left(\frac{1}{\gamma} u\right),$$

by means of the Fenchel Conjugate $F^*(\hat{x}) = -\inf_x \{f(x) - x \cdot \hat{x}\}$ [3]. Going back to (4), it is easy to see that for $f(z) = \frac{1}{2}\|z - w\|_2^2$, we have $f^*(s) = \frac{1}{2}s \cdot s + s \cdot w$. The conjugate of the $\ell_{2,1}$ norm can be derived using the definition of Fenchel Conjugate and the conjugate of the ℓ_2 norm (the indicator function of the unitary ball), obtaining that $r^*(s)$ is the indicator function of the unitary balls for each group, $\iota_{\bigwedge_{n=1}^{N-1} \|s_n\|_2 \leq 1}$. Therefore, the dual problem becomes:

$$\min_u \left\{ \frac{1}{2}\|\bar{D}^\top u\|_2^2 - u^\top \bar{D} w + \iota_{\bigwedge_{n=1}^{N-1} \|u_n\|_2 \leq \gamma} \right\} \equiv \begin{array}{l} \min\limits_u \left\{ \frac{1}{2}\|\bar{D}^\top u - w\|_2^2 \right\} \\ \text{s.t. } \|u_n\|_2 \leq \gamma, \; 1 \leq n \leq N-1, \end{array} \tag{5}$$

where we have completed squares and changed the indicator function to a set of constraints. Since problem (5) is quadratic with simple convex constraints, it can be easily solved using projected gradient. After that, z_w (i.e., the result of the proximal operator) can be recovered from the dual solution u^* through the equality $z_w = w - \bar{D}^\top u^*$, which follows from $0 = \nabla_z \mathcal{L} = z_w - w + B^\top u^*$.

To finish this section, we observe that the form of the $\ell_{2,1}$ norm implicitly assumes a 1-dimensional spatial structure for the data. However, many problems of interest, such as image processing, present a natural multidimensional structure that cannot be captured by the $\ell_{2,1}$ penalty. Working only with the GTV penalty, and as in [2], a solution for this is to combine several 1-dimensional GTV penalties to obtain a multidimensional GTV. For example, for problems with a 2-dimensional structure, we penalize changes in both row and column-wise adjacent features. More precisely, denoting the i-th row by $w^{[i,\cdot]}$ and the j-th column by $w^{[\cdot,j]}$, we can define the 2-dimensional GTV regularizer as $\mathrm{GTV}^{2d}(w) = \sum_i \mathrm{GTV}\left(w^{[i,\cdot]}\right) + \sum_j \mathrm{GTV}\left(w^{[\cdot,j]}\right)$. This can be easily extended to more than two dimensions but, again, notice that this multidimensional GTV regularizer is the sum of 1-dimensional GTVs. Those corresponding to the same dimension (for example, the terms $\mathrm{GTV}\left(w^{[i,\cdot]}\right)$ corresponding to the different columns) apply over different variables, and are therefore separable, so the proximal operator of the summation of a particular dimension can be computed just by composing the individual proximal operators. Nevertheless, each complete summation applies over all the variables, and they cannot be separated. In order to combine the proximal operators of the different dimensions we can use once again the PD algorithm. Similarly, for the case of a complete multidimensional GFL linear model, we should use PD to compute the proximal operator of the multidimensional GTV regularizer, and then combine the GTV and GL proximal operators applying again PD.

3 Experiments

We will present next an application of the GFL model over a synthetic regression example and the use of the GTV regularizer for colour image denoising.

We consider first a synthetic structured linear problem where pattern features are divided into 100 3-dimensional groups, i.e., we have $N = 100$ and $V = 3$. The optimal weights are structured in 4 consecutive segments of 25 groups with constant values for the three group coordinates. This defines an optimal weight $w^* = (w_1^*, w_2^*, w_3^*, w_4^*)^\top$ with each w_i^* constant; w^* is thus built in such a way that it makes the features to be simultaneously either active or inactive and in such a way that adjacent features have a block behaviour. The optimal w^* is then perturbed to obtain a weight vector of the form $\tilde{w}_{n,v} = w_{n,v}^* + \eta_{n,v}$ with $\eta \sim \mathcal{N}(0, 0.1)$ Gaussian noise. Random independent patterns x^p are then generated by a $\mathcal{N}(0,1)$ distribution, and the values $y^p = \tilde{w} \cdot x^p + \hat{\eta}_p$ with $\hat{\eta} \sim \mathcal{N}(0, 0.1)$ then define a regression problem. Notice that the underlying spatial structure of the weights imposes also an spatial structure on the y^p values. Moreover, if the number of generated x patterns is well below the problem dimension of 300, we will end up with an ill-posed problem. We will consider 600, 300, 100 and 50 training patterns and solve the regression problem using LA, GL, FL and GFL. In the latter case, we apply the complete 1-dimensional GFL linear model (with both the 1-dimensional GTV and the GL terms). The corresponding regularization parameters are chosen so that the estimated weights are closest to the generating weights in the ℓ_1 distance. Table 3 presents the corresponding results in terms of the distances $\|w - w^*\|_1$ and $\|w - w^*\|_2$. As can be seen,

Table 1. Distance to optimal weights for the considered structured linear regression models as a function of the number of training samples (lower is better)

Mod	**Training Size**			
	600	**300**	**100**	**50**
LA	23.59	29.91	1016.60	1284.88
GL	23.70	30.75	1024.45	1304.23
FL	10.61	11.28	*13.88*	29.60
GFL	*9.35*	*10.93*	15.57	*26.43*

$\|w - w^*\|_1$

Mod	**Training Size**			
	600	**300**	**100**	**50**
LA	1.74	2.21	96.28	126.47
GL	1.76	2.26	92.25	128.76
FL	0.86	0.97	1.26	2.40
GFL	*0.72*	*0.92*	*1.24*	*2.05*

$\|w - w^*\|_2$

GFL achieves the lowest $\|w - w^*\|_1$ distance in all the cases but one, and the lowest $\|w - w^*\|_2$ for all of them. Only FL is comparable, whereas LA and GL values are clearly worse for the 600 and 300 pattern problems and markedly fail when used with few training samples. As reference value, observe that the distances of the perturbed weights to the original ones are $\|\tilde{w} - w^*\|_1 = 24.81$ and $\|\tilde{w} - w^*\|_2 = 1.78$, close to the FL and GFL values but far away from the LA, GL ones. Moreover, Fig. 3 shows how GFL recovers quite well the inherent structure of the problem.

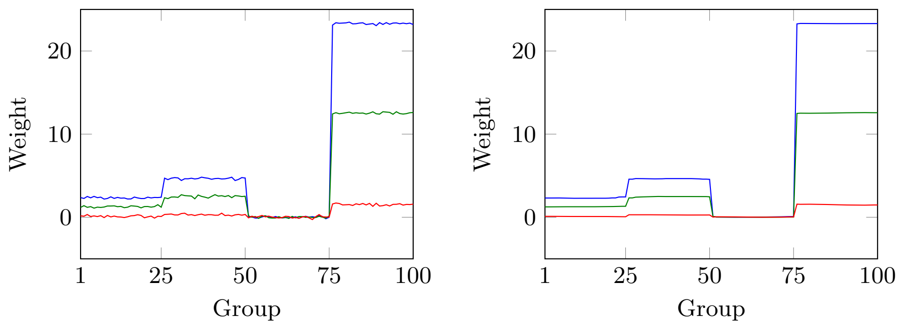


Fig. 1. Noisy weights (left) and weights recovered by GFL (right), using 600 patterns. The three colours represent different variables of the same group.

We consider next how to apply GTV to denoise colour images. Notice that images have a natural spatial structure, as pixels change smoothly and can be considered nearly constant in nearby regions (except in objects borders). Therefore, TV regularization has been extensively used for this task [5] on gray level images, in the form of the denoising model $\min_I \frac{1}{2}\|I - \tilde{I}\|_2^2 + \mathrm{TV}^{2d}(I)$ for a noisy image $\tilde{I}$ and some bidimensional form of TV, whose block structure permits to preserve the borders. When dealing with colour images a possible option is to apply TV denoising independently to each of the three RGB layers. However, we can also consider each pixel as a multi-valued (R,G,B) feature, making GTV fit naturally into this problem using the whole of the problem structure. Specifically, we will use the 2-dimensional GTV proximal operator, which can be easily computed as explained in Sect. 2. We will work with two different colour images. The first one (*peppers*) is perturbed by additive noise as $\tilde{I} = I + n$, with I the original image and $n \sim \mathcal{N}(0, 0.05)$. For the second image (*Lena*) we consider speckle noise, i.e., multiplicative uniform noise, with $\tilde{I} = I + uI$, where u is uniform with 0 mean and variance 0.25. Our goal here is to compare the potential advantages of GTV over 2-dimensional TV and for each model we select the optimal GTV and TV penalties as the ones that give the best Improved Signal-to-Noise Ratio (ISNR) over a single perturbed sample for each image. We then test TV and GTV denoising over 10 other different perturbations for additive and multiplicative noise. In all cases GTV performed better than TV, yielding an average ISNR of 10.73 ± 0.36 for additive noise and of 12.24 ± 0.24 for multiplicative noise; on the other hand, the ISNR averages for TV are 8.68 ± 0.27 and 10.97 ± 0.41, respectively. Figure 3 contains an example of denoising for the two different image and noise models described above.

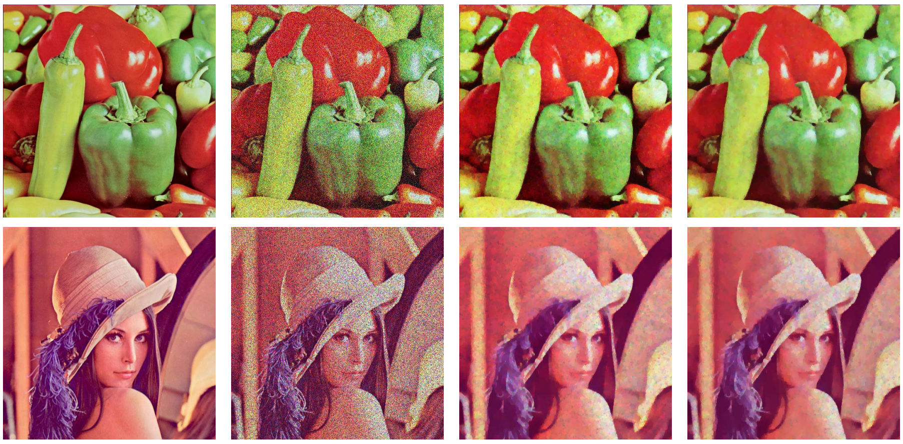


Fig. 2. Denoising with additive (upper row) and multiplicative (lower row) noise

4 Conclusions

In this work we have proposed the Group Total Variation (GTV) regularizer, combining the multidimensional group-sparse features of the Group Lasso regularizer with the block spatial structure of the Total Variation penalty used by Fused Lasso. The GTV regularizer thus appears as a useful tool to reconstruct multidimensional patterns with a spatial structure that reflects smooth changes along the group features. Colour image denoising fits nicely in this framework and we have shown that GTV performs better than applying 1-dimensional Total Variation independently on each colour. Moreover, this GTV regularizer can be merged with a Group Lasso (GL) term, leading to what we call Group Fused Lasso (GFL). We have illustrated over a synthetic example how GFL effectively captures block structure when present, and makes use of it to address linear ill-posed problems with a number of features much larger than the sample size.

This kind of spatial structure can be found in other real world problems, particularly those for which the underlying data features are associated to geographical locations. Any sensible linear regression models for such problems should assign similar weight values to spatially close features, which is exactly the behaviour that GFL enforces. As further work we intend to study the advantages of GFL in such a kind of problems, which will require the use of the complete 2-dimensional GFL model as explained at the end of Sect. 2, and also to analyse the numerical complexity of the proposed models and possible ways to improve it.

Acknowledgement. With partial support from Spain's grant TIN2010-21575-C02-01 and the UAM–ADIC Chair for Machine Learning. The first author is supported by the FPU–MEC grant AP2008-00167.

References

1. Bach, F., Jenatton, R., Mairal, J., Obozinski, G.: Convex Optimization with Sparsity-Inducing Norms (2011), `http://www.di.ens.fr/~fbach/opt_book.pdf`
2. Barbero, A., Sra, S.: Fast newton–type methods for total variation regularization. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), New York, NY, USA, pp. 313–320 (2011)
3. Bauschke, H., Combettes, P.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces. Springer (2011)
4. Beck, A., Teboulle, M.: A fast iterative shrinkage–thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences 2(1), 183–202 (2009)
5. Bioucas-Dias, J.M., Figueiredo, M.A.T.: A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration. IEEE Transactions on Image Processing 16(12), 2992–3004 (2007)
6. Bleakley, K., Vert, J.P.: The group fused Lasso for multiple change-point detection. ArXiv e-prints (2011)
7. Combettes, P.L., Pesquet, J.C.: Proximal splitting methods in signal processing. Recherche 49, 1–25 (2009)
8. Tibshirani, R.: Regression shrinkage and selection via the Lasso. J. Roy. Statist. Soc. Ser. B 58(1), 267–288 (1996)
9. Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., Knight, K.: Sparsity and smoothness via the fused lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67(1), 91–108 (2005)
10. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society – Series B: Statistical Methodology 68(1), 49–67 (2006)

Exponential Synchronization of a Class of RNNs with Discrete and Distributed Delays*

Farouk Chérif[1], Hajer Brahmi[2], Boudour Ammar[2], and Adel M. Alimi[2]

[1] ISSATS, Laboratory of Math Physics; Specials Functions and Applications, LR11ES35, Ecole Supérieure des Sciences et de Technologie, 4002- Sousse- Tunisia
[2] REGIM: REsearch Groups on Intelligent Machines, University of Sfax, National Engineering School of Sfax (ENIS), BP 1173, Sfax, 3038, Tunisia
faroukcheriff@yahoo.fr, {boudour.ammar,adel.alimi}@ieee.org

Abstract. This paper studies the exponential synchronization of RNNs. The investigations are carried out by means of Lyapunov stability method and the Halanay inequality lemma. Finally, a numerical example with graphical illustrations is given to illuminate the presented synchronization scheme.

Keywords: Recurrent Neural Networks, Exponential synchronization, Stability.

1 Introduction

In the last decade, there has been increasing interest in exploring of recurrent neural networks (RNNs) since they have a wide range of applications, for instance, signal processing, pattern recognition, associative memory and combinatorial optimization. In particular, different types of recurrent neural networks (HNNs, CNNs) have been used and applied to study the qualitative properties such as existence and oscillations of solutions ([2], [4], [5]). Hence, there have been extensive results on the problem of the existence and synchronization of RNNs with constant time delays and time-varying delays in the literature. However, there exist few results on the dynamical behaviors of RNNs with continuously distributed delays. In particular, exponential synchronization of RNNs is of paramount importance in a variety of complex physical, chemical, and biological systems [13]. It is well known that such synchronization strategies have potential applications in several areas such as secure communication ([11], [14]) biological oscillators [3] and animal gaits [7]. It should be mentioned that there are different notions of synchronization, such as phase synchronization [16], generalized synchronization [17], lag synchronization [18], and identical synchronization [15]. In this paper, motivated by the above discussions, we are concerned with the exponential synchronization of a class of recurrent neural networks with varying-time

* This work is supported in part by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

© Springer-Verlag Berlin Heidelberg 2013

coefficients and mixed delays. Thus, the goal in this paper is to design an appropriate controller such that the coupled neural networks remain synchronized. This paper is organized as follows. In Section 2, the synchronization problem to be considered is formulated. In Section 3, a new sufficient condition for the exponential synchronization is obtained. In Section 4, numerical simulations is given to show the validity of theoretical result.

2 Exponential Synchronization Problem

The model of the delayed recurrent neural network considered in this paper is described by the following state equations

$$\begin{aligned} \dot{x_i}\left(t\right) &= -a_i x_i(t) + \sum_{j=1}^{n} c_{ij}\left(t\right) f_j(x_j\left(t\right)) + \sum_{j=1}^{n} d_{ij}\left(t\right) f_j(x_j\left(t-\tau\right)) \\ &+ \sum_{j=1}^{n} p_{ij}\left(t\right) \int_{t-\sigma}^{t} f_j(x_j(s))ds + J_i\left(t\right), \\ x_i(t) &= \psi_i(t), \varrho \le t \le 0, 1 \le i \le n, \end{aligned} \tag{1}$$

where n is the number of the neurons in the neural network, $x_i(t)$ denotes the state of the ith neural neuron at time t, $f_j(x_j(t))$ is the activation function of jth neuron at time t. The functions $c_{ij}\left(\cdot\right)$, $d_{ij}\left(\cdot\right)$ and $p_{ij}\left(\cdot\right)$ denote, respectively, the connection weights, the discretely delayed connection weights, and the distributively delayed connection weights, of the jth neuron on the i neuron. $J_i\left(\cdot\right)$ is the external bias on the ith neuron, a_i denotes the rate with which the ith neuron will reset its potential to the resting state in isolation when disconnected from the network and external inputs. τ is the constant discrete time delay and $\varrho = \max\left(\tau, \sigma\right)$.

Now let us give the following notations and concepts used throughout this paper. For $x \in \mathbb{R}^n$, let $\|x\| = \left(x^T x\right)^{\frac{1}{2}} = \left(\sum_{j=1}^{n} x_i^2\right)^{\frac{1}{2}}$ denote the Euclidean vector norm, and for a matrix $A \in \mathcal{M}_n\left(\mathbb{R}\right)$, let $\|A\|$ indicate the norm of A induced by the Euclidean vector norm, i.e.,$\|A\| = \left(\lambda_{\max}\left(A^T A\right)\right)^{\frac{1}{2}}$, where $\lambda_{\max}\left(A\right)$ represents the maximum eigenvalue of matrix A and T denotes the transpose of a matrix. We denote a vector solution of the above system as $x(t) = (x_1(t), x_2(t), \ldots, x_n(t))^T$. The neural network (1) can be rewritten in the following matrix-vector form

$$\begin{aligned} \dot{x}\left(t\right) &= -Dx(t) + Cf(x\left(t\right)) + Df(x\left(t-\tau\right)) + P\int_{t-\sigma}^{t} f(x(s))ds + J\left(t\right) \\ x(t) &= \psi(t), \varrho \le t \le 0. \end{aligned} \tag{2}$$

Throughout this paper, we make the following assumptions:

(H_1) For all $1 \le j \le n$, there exist positive constant numbers $L_j > 0$ such that for all $x, y \in \mathbb{R}$

$$\left|f_j(x) - f_j(y)\right| < L_j\left|x - y\right|,$$

(H_2) For all $1 \leq i \leq n$, $a_i > 0$ and $\tau, \sigma > 0$,

Let us introduce the following controlled slave (or response) system:

$$\begin{array}{rcl} \dot{z}_i(t) &=& -a_i z_i(t) + \sum\limits_{j=1}^{n} c_{ij}(t) f_j(z_j(t)) + \sum\limits_{j=1}^{n} d_{ij}(t) f_j(z_j(t-\tau)) \\ + && \sum\limits_{j=1}^{n} p_{ij}(t) \int\limits_{t-\sigma}^{t} f_j(z_j(s))ds + J_i(t) + u_i \\ z_i(t) &=& \varphi_i(t), \varrho \leq t \leq 0, 1 \leq i \leq n, \end{array} \tag{3}$$

in which $u_i(t)$ denotes the external control input that will be appropriately designed for an certain control objective.

3 Exponential Synchronization of the RNNs

Definition 1. *The systems* (1) *and the uncontrolled system* (2) *(i.e.* $u_i = 0, \forall 1 \leq i \leq n$ *in* (3)*) are said to be exponentially synchronized if there exist constants* $\eta \geq 1$ *and* $\alpha > 0$ *such that*

$$|x_i(t) - z_i(t)| \leq \eta |x_i(0) - z_i(0)| e^{-\alpha t}$$

for any $t \geq 0$*. Moreover, the constant* α *is defined as the exponential synchronization rate.*

From (1) and (3), the following error dynamics equation can be obtained:

$$\begin{array}{rl} \dot{e}_i(t) = & -a_i e_i(t) + \sum\limits_{j=1}^{n} c_{ij}(t) F_j(e_j(t)) + \sum\limits_{j=1}^{n} d_{ij}(t) F_j(e_j(t-\tau)) \\ + & \sum\limits_{j=1}^{n} p_{ij}(t) \int\limits_{t-\sigma}^{t} F_j(e_j(s))ds + u_i, 1 \leq i \leq n, \end{array} \tag{4}$$

where $e(t) = x(t) - z(t)$ is the error term, and $F(e(t)) = f(x(t)) - f(z(t))$; $F(e(t-\tau)) = f(x(t-\tau)) - f(z(t-\tau))$.

As long as the control input stabilize the system, the error vector $e(t)$ converges to zero as time t goes to infinity i.e. $\lim\limits_{t\to+\infty} e(t) = \lim\limits_{t\to+\infty} x(t) - z(t) = 0$. If the state variables of the drive system are used to drive the response system, then the control input vector with state feedback is designed as follows:

$$\begin{pmatrix} u_1(t) \\ \vdots \\ u_n(t) \end{pmatrix} = M \begin{pmatrix} x_1(t) - z_1(t) \\ \vdots \\ x_n(t) - z_n(t) \end{pmatrix} = M \begin{pmatrix} e_1(t) \\ \vdots \\ e_n(t) \end{pmatrix} \tag{5}$$

where $M = (m_{ij})_{n\times n}$ is the controller gain matrix and will be appropriately chosen for exponentially synchronizing both drive system and response system. It follows that the error dynamics can be expressed by the following compact form:

$$\dot{e}(t) = -Ae(t) + CF(e(t)) + DF(e(t-\tau)) + P\int\limits_{t-\sigma}^{t} F(e(s))ds + u$$

Lemma 1. *([1]) For all $(n \times n)$ real symmetric matrix M, one has M is positive definite if and only if all its eigenvalues are positive. Furthermore, for all $x \in \mathbb{R}^n$*

$$\lambda_{\min}(M)\|x\|^2 \leq x^T M x \leq \lambda_{\max}(M)\|x\|^2$$

where $\lambda_{\min}(M)$ ($\lambda_{\max}(M)$) represents the minimum (resp. the maximum) eigenvalue of the matrix M.

Lemma 2. *(Halanay inequality lemma [9]). Let $\rho \geq 0$ be a constant, and $V(\cdot)$ be a non-negative continuous function defined for $[-\rho, +\infty[$ which satisfies*

$$\dot{V}(t) \leq -pV(t) + q\left(\sup_{t-\rho \leq s \leq t} V(s)\right)$$

for $t \geq 0$, where p and q are constants. If $p > q > 0$, then

$$V(t) \leq \left(\sup_{-\rho \leq s \leq 0} V(s)\right) e^{-\delta t}$$

for $t > 0$, where δ is a unique positive root of the equation $\delta = p - qe^{\delta\tau}$.

Theorem 1. *Suppose that the conditions $(H_1)-(H_2)$ hold. If the controller gain matrix M in (5) is real symmetric and positive definite satisfying*

$$\frac{\max\limits_{1\leq i\leq n} L_i\left(2\|C\| + \|D\| + \sigma\|P\|\right)}{2\min\limits_{1\leq i\leq n} a_i + 2\lambda_{\min}(M)} < 1, (H_3)$$

then the exponential error system (4) converges exponentially.

Proof. First, it is clear that in view of (H_1)

$$\begin{aligned}\|F(e(t-\tau))\|^2 &= \sum_{i=1}^{n} F_i^2(e(t-\tau)) \leq \sum_{i=1}^{n} L_i^2(e_i^2(t-\tau)) \\ &\leq \max_{1\leq i\leq n} L_i^2 \|e(t-\tau))\|^2 .\end{aligned}$$

Similarly, $\|F(e(t))\| \leq \max\limits_{1\leq i\leq n} L_i^2 \|e(t))\|^2$. In order to confirm that the origin of (4) is globally exponential synchronization, let us consider the continuous function, V defined as follows: $V(t) = \frac{1}{2}e(t)^T e(t) = \frac{1}{2}\|e\|^2$. Calculating the time derivative of V along the trajectory by using the vector norm, matrix norm and from the inequalities above, we obtain immediately

$$\begin{aligned}\dot{V}(t) = &-e(t)^T Ae(t) + e(t)^T CF(e(t)) + e(t)^T DF(e(t-\tau)) \\ &+e(t)^T P\int\limits_{t-\sigma}^{t} K(t-s)F(e(s))ds - e(t)^T Me(t)\end{aligned}$$

$$\leq -\sum_{i=1}^{n} a_i e_i^2 + \|e\| \|C\| \|F(e(t))\| + \|e\| \|D\| \|F(e(t-\tau))\|$$
$$+ \|e\| \|P\| \int_{t-\sigma}^{t} \|F(e(s))\| ds - \lambda_{\min}(M) \|e(t)\|^2$$

By Cauchy Shwartz inequality one can obtain

$$\dot{V}(t) \leq -\min_{1\leq i\leq n} a_i \|e\|^2 + \|e(t)\| \|C\| \max_{1\leq i\leq n} L_i \|e(t))\| + \max_{1\leq i\leq n} L_i \|e(t)\| \|D\| \|e(t-\tau))\|$$
$$+ \max_{1\leq i\leq n} L_i \|e(t)\| \|P\| \left(\int_{t-\sigma}^{t} ds\right)^{\frac{1}{2}} \left(\int_{t-\sigma}^{t} \|e(s)\|^2 ds\right)^{\frac{1}{2}} - \lambda_{\min}(M) \|e(t)\|^2$$
$$\leq \left(-\min_{1\leq i\leq n} a_i + \|C\| \max_{1\leq i\leq n} L_i - \lambda_{\min}(M)\right) \frac{1}{2} \|e(t)\|^2 + \max_{1\leq i\leq n} L_i \|e(t)\| \times$$
$$\times \|D\| \|e(t-\tau))\| + \max_{1\leq i\leq n} L_i \|e(t)\| \|P\| \sqrt{\sigma} \left(\int_{t-\sigma}^{t} \|e(s)\|^2 ds\right)^{\frac{1}{2}}$$
$$\leq \left(-\min_{1\leq i\leq n} a_i + \|C\| \max_{1\leq i\leq n} L_i - \lambda_{\min}(M)\right) \|e(t)\|^2 + \frac{1}{2} \|D\| \times$$
$$\times \quad \max_{1\leq i\leq n} L_i \left(\|e(t)\|^2 + \|e(t-\tau)\|^2\right) + \sqrt{\sigma} \max_{1\leq i\leq n} L_i \|e(t)\| \|P\| \sqrt{\sigma} \left(\max_{t-\sigma\leq s\leq t} \|e(s)\|^2\right)^{\frac{1}{2}}$$
$$\leq \left(-\min_{1\leq i\leq n} a_i + \|C\| \max_{1\leq i\leq n} L_i - \lambda_{\min}(M)\right) \|e(t)\|^2 + \frac{1}{2} \|D\| \max_{1\leq i\leq n} L_i \times$$
$$\times \quad \left(\|e(t)\|^2 + \|e(t-\tau)\|^2\right) + \frac{\sigma}{2} \max_{1\leq i\leq n} L_i \|P\| \left(\|e(t)\|^2 + \max_{t-\sigma\leq s\leq t} \|e(s)\|^2\right)$$
$$\leq \left(-2\min_{1\leq i\leq n} a_i + 2\|C\| \max_{1\leq i\leq n} L_i + \|D\| \max_{1\leq i\leq n} L_i - 2\lambda_{\min}(M) + \sigma \max_{1\leq i\leq n} L_i \|P\|\right) V(t)$$
$$+ \frac{1}{2} \|D\| \max_{1\leq i\leq n} L_i \|e(t-\tau))\|^2 + \frac{\sigma}{2} \max_{1\leq i\leq n} L_i \|P\| \max_{t-\sigma\leq s\leq t} \|e(s)\|^2$$
$$\leq -\left(2\min_{1\leq i\leq n} a_i - 2\|C\| \max_{1\leq i\leq n} L_i - \|D\| \max_{1\leq i\leq n} L_i + 2\lambda_{\min}(M) - \sigma \max_{1\leq i\leq n} L_i \|P\|\right) V(t)$$
$$+ \max_{1\leq i\leq n} L_i \left(\|D\| + \frac{\sigma}{2} \|P\|\right) \max_{t-\rho\leq s\leq t} V(s)$$

Now, in virtue of lemma 1 and (H_3) it follows that $V(t) \leq \left(\sup_{-\rho\leq s\leq t0} V(s)\right) e^{-\delta t}$ where

$$\delta = \left(2\min_{1\leq i\leq n} a_i - \max_{1\leq i\leq n} L_i (2\|C\| + \|D\| + \sigma \|P\|) + 2\lambda_{\min}(M)\right)$$
$$- \max_{1\leq i\leq n} L_i \left(\|D\| + \frac{\sigma}{2} \|P\|\right) e^{\delta\rho}.$$

Therefore, $V(e(t))$ converges to zero exponentially, which in turn implies that $e(t)$ also converges globally and exponentially to zero with a convergence rate of $\frac{\delta}{2}$, i.e.$\|e(t)\| \leq \left(\sup_{-\rho\leq s\leq s} \|\phi(s) - \psi(s)\|\right) e^{-\delta \frac{t}{2}}$.

In other words, every trajectory $z_i(t)$ of (3) must synchronize exponentially toward the $x_i(t)$ with a convergence rate of $\frac{\delta}{2}$. This completes the proof.

Remark 1. Clearly and from the above study, the sufficient condition for exponential synchronization of systems (1) and (3) depends only on the continuous delay but relies on the connection weights and the controller gain. Besides, when for all $1 \leq i, j \leq n$, $p_{ij} = 0$ and $J(\cdot)$ is constant, model (1) and (2) in this paper become the models in [12]. On the other hand, in [6] under similar hypothesis, authors derive exponential synchronization criteria for two chaotic neural networks under the configuration of the master slave mode by applying the Lyapunov stability approach and the Halanay inequality. However, the conditions of Theorem 1 in this paper is easy to test in practice. So, the results in [6] is a special case of the results in this paper. It should be mentioned that the method in this paper is not as same as the method in [8] and [10].

4 An Illustrative Example

In order to illustrate some feature of our main results, in this section, we will apply our main results to some special three-dimensional systems and demonstrate the efficiencies of our criteria.

Example 1. Let us consider the following delayed recurrent neural network

$$\begin{aligned}\dot{x_i}(t) = -a_i x_i(t) + \sum_{j=1}^{3} c_{ij}(t) f_j(x_j(t)) + \sum_{j=1}^{3} d_{ij}(t) f_j(x_j(t-\tau)) \\ + \sum_{j=1}^{3} p_{ij}(t) \int_{t-\sigma}^{t} f_j(x_j(s))ds + J_i(t),\end{aligned}$$

and the response recurrent neural network is designed as follows:

$$\begin{aligned}\dot{z_i}(t) = -a_i z_i(t) + \sum_{j=1}^{3} c_{ij}(t) f_j(z_j(t)) + \sum_{j=1}^{3} d_{ij}(t) f_j(z_j(t-\tau)) \\ + \sum_{j=1}^{3} p_{ij}(t) \int_{t-\sigma}^{t} f_j(z_j(s))ds + J_i(t) + u_i(t)\end{aligned}$$

$$\dot{e}(t) = -Ae(t) + CF(e(t)) + DF(e(t-\tau)) + P\int_{t-\sigma}^{t} F(e(s))ds + u$$

Pose: $a_1 = 11, a_2 = 17, a_3 = 13, f_j(x) = x, \tau = 1, \sigma = 2$ and

$$C = \begin{pmatrix} 1 & -3 & -2 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \end{pmatrix}, D = \begin{pmatrix} 2 & -3 & -1 \\ 0 & -1 & 4 \\ -1 & 0 & 2 \end{pmatrix}, P = \begin{pmatrix} 0.5 & -1.5 & 1 \\ 1 & 0 & 2 \\ 2 & -0.5 & 1 \end{pmatrix}, M = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

So the condition (H_3) is satisfied since

$$\frac{\max\limits_{1\leq i\leq 3} L_i\,(2\times 3.\,840\,8+4.\,686\,2+2\times 3.\,312\,6)}{2\times 9+2\times \lambda_{\min}\,(M)} = 0.86332 < 1$$

By using the matlab Toobox, one can obtain the graphical illustration Fig. 1

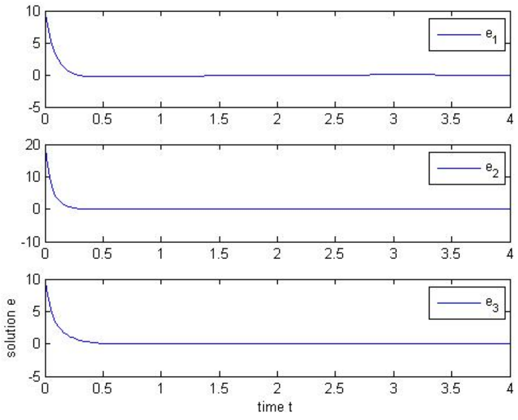


Fig. 1. The exponential synchronization error

5 Conclusion

By Constructing an appropriate linear feedback controller, this paper addresses the problem of exponential synchronization of a class of recurrent neural networks with mixed delays. Based on the properties of a recurrent attractor, we gave a new synchronization criterion for the considered system by using Lyapunov method and the well known Halanay lemma. To demonstrate the effectiveness of the proposed method, a numerical example is used.

References

1. Allen, L., Bridges, T.J.: Numerical exterior algebra and the compound matrix method. Numerische Mathematik 92, 197–232 (2002)
2. Ammar, B., Chérif, F., Alimi, M.A.: Existence and Uniqueness of Pseudo Almost Periodic Solutions of Recurrent Neural Networks with Time-Varying Coefficients and Mixed Delays. IEEE Transactions on Neural Networks and Learning Systems 23(1), 109–118 (2012)
3. Bazhenov, M., Huerta, R., Rabinovich, M.I., Sejnowski, T.: Cooperative behavior of a chain of synaptically coupled chaotic neurons. Phys. D 116, 392–400 (1998)
4. Cao, J.D., Wang, J.: Global exponential stability and periodicity of recurrent neural networks with time delay. IEEE Trans. Circ. Syst. I 52(5), 920–931 (2005)

5. Chérif, F.: Existence and global exponential stability of pseudo almost periodic solution for SICNNs with mixed delays. JAMC 39(1) (2011)
6. Cheng, C.-J., Liao, T.-L., Hwang, C.-C.: Exponential synchronization of a class of chaotic neural networks. Chaos, Solitons and Fractals 24, 197–206 (2005)
7. Collins, J.J., Stewart, I.N.: Coupled nonlinear oscillators and the symmetries of animal gaits. J. Nonlinear Sci. 3, 349–392 (1993)
8. Cui, B., Lou, X.: Synchronization of chaotic recurrent neural networks with time-varying delays using nonlinear feedback control. Chaos, Solitons and Fractals 39, 288–294 (2009)
9. Gopalsamy, K.: Stability and oscillations in delay differential equations of population dynamics. Kluwer Academic Publishers, The Netherlands (1992)
10. Huang, T., Chen, G., Kurths, J.: Synchronization of chaotic systems with time-varying coupling delays. Discrete and Continuous Dynamical Systems, Series B 16(4) (2011)
11. Itoh, M., Wu, C.W., Chua, L.O.: Communication systems via chaotic signals from a reconstruction viewpoint. Int. J. Bifur. Chaos Appl. Sci. Eng. 7, 275–286 (1997)
12. Lu, H., van Leeuwen, C.: Synchronization of chaotic neural networks via output or state coupling. Chaos, Solitons and Fractals 30, 166–176 (2006)
13. Ott, E., Grebogi, C., Yorke, J.A.: Controlling chaos. Rev. Lett. 64, 1196–1199 (1990)
14. Parlitz, U., et al.: Transmission of digital signals by chaotic synchronization. Int. J. Bifur. Chaos Appl. Sci. Eng. 2, 973–977 (1992)
15. Pecora, L.M., Carroll, T.L.: Synchronization in chaotic systems. Phys. Rev. Lett. 64, 821–824 (1990)
16. Rosenblum, M.G., Pikovsky, A.S., Kurths, J.: Phase synchronization of chaotic oscillators. Phys. Rev. Lett. 76, 1804–1807 (1996)
17. Rulkov, N.F., Sushchik, M.M., Tsimring, L.S., Abarbanel, H.D.I.: Generalized synchronization of chaos in directionally coupled chaotic systems. Phys. Rev. E 51, 980–994 (1995)
18. Rosenblum, M.G., Pikovsky, A.S., Kurths, J.: From phase to lag synchronization in coupled chaotic oscillators. Phys. Rev. Lett. 78, 4193–4196 (1997)

Variational Foundations of Online Backpropagation

Salvatore Frandina, Marco Gori, Marco Lippi, Marco Maggini, and Stefano Melacci

Department of Information Engineering and Mathematical Sciences, University of Siena, Italy
{frandina,marco,lippi,maggini,mela}@diism.unisi.it

Abstract. On-line Backpropagation has become very popular and it has been the subject of in-depth theoretical analyses and massive experimentation. Yet, after almost three decades from its publication, it is still surprisingly the source of tough theoretical questions and of experimental results that are somewhat shrouded in mystery. Although seriously plagued by local minima, the batch-mode version of the algorithm is clearly posed as an optimization problem while, in spite of its effectiveness, in many real-world problems the on-line mode version has not been given a clean formulation, yet. Using variational arguments, in this paper, the on-line formulation is proposed as the minimization of a classic functional that is inspired by the principle of minimal action in analytic mechanics. The proposed approach clashes sharply with common interpretations of on-line learning as an approximation of batch-mode, and it suggests that processing data all at once might be just an artificial formulation of learning that is hopeless in difficult real-world problems.

Keywords: on-line Backpropagation, principle of least action, regularization, local minima, dissipative systems.

1 Introduction

In classical statistics, sum-minimization problems arise in least squares and in maximum-likelihood estimation (for independent observations). The general class of estimators that arise as minimizers of sums are called M-estimators. Backpropagation [8] was proposed to efficiently compute the gradient of the cost function associated with a supervised neural network. Importance traces of the idea behind the algorithm can be found mostly in [9], but also in [2]. In spite of the plain numerical computation of the gradient, in many cases, it makes it possible to break the barrier that enables many applications of neural networks to real-world problems. Unfortunately, the convergence of the algorithm is seriously plagued by the presence of local minima in the error function [5]. In many cases, instead of performing a classic gradient descent scheme, the gradient computation for single examples (on-line mode) has been profitably used by updating directly the parameters, without accumulating those contributions for all the training set. The on-line scheme is especially adequate to real-world

© Springer-Verlag Berlin Heidelberg 2013

Table 1. Links between machine learning and analytic mechanics

Links with Analytic Mechanics		
variable	machine learning	analytic mechanics
w_i	weight	particle position
$\dot{w}_i$	weight variation	particle velocity
V	loss temporal derivative	potential energy
T	temporal smoothness	kinetic energy
$\mathcal{L} = T - V$	Cognitive Lagrangian	Mechanical Lagrangian
$\mathcal{S} = \int_0^{t_e} \mathcal{L}\ dt$	Cognitive Action	Mechanical Action

problems where the examples are streamed continuously in time. There is plenty of evidence that such a stochastic gradient descent has been very effective in the case of large-scale problems [1]. Amongst others, the Backpropagation on-line training scheme is often regarded as a way to get around shallow local minima of the cost function, but like for the batch-mode scheme, it is quite hard to understand the conditions of convergence, apart from relatively simple cases [4].

After almost three decades from its publication, on-line Backpropagation is still surprisingly the source of tough theoretical questions, and it has not received a fully satisfactory formulation, yet. Using variational arguments, in this paper, the on-line formulation is proposed as the minimization of a classic functional that is inspired by the principle of least action in analytic mechanics. However, the classic Lagrangian is replaced with a *time-variant* function that is responsible of a dissipative behavior that plays a major role in any learning process. We prove that a "strong dissipation" transforms the continuous time differential law coming from the Euler-Lagrange equation into the classic on-line Backpropagation with its stochastic gradient numerical computation. The proposed approach clashes sharply with common interpretations of on-line learning as an approximation of batch-mode. On the other hand, differently from what is generally assumed, it suggests that processing data all at once might be just an artificial formulation of learning that is hopeless in difficult real-world problems.

2 On-Line Backpropagation Revisited

We consider a feedforward neural network as a function which transforms a given input $x \in I\!R^d$ into a real number, that is $f : (x, w) \in \mathcal{X} \times \mathcal{W} \to I\!R$, being w the vector of weights and x the input. The analysis carried out in this paper does not make any hypothesis on the network structure and, consequently, on $f(x, w)$, but we like to think of it in terms of feedforward networks mostly because of their universal approximation capabilities and their biological inspiration [7].

POTENTIAL ENERGY

Now we introduce the loss function $\overline{V}(f, y)$, along with the set of supervised pairs $\mathcal{P} = \{(x_\kappa, y_\kappa)\}_{\kappa=1}^{\ell}$. For example, $\overline{V}(f, y)$ can be the hinge function – typical for classification – or the quadratic function $(f(x_\kappa) - y_\kappa)^2$ – typical of regression. Let ζ be a mollifier. As an example, we can choose

$$\zeta_\epsilon(\tau) = \begin{cases} Z_\epsilon \cdot \exp\left(1 - \epsilon^2/(\epsilon^2 - \tau^2)\right) & \text{if } |\tau| < \epsilon \\ 0 \ \text{ if } |\tau| \geq \epsilon, \end{cases}$$

where Z_ϵ is taken in such a way that $\int_{-\infty}^{+\infty} \zeta(\tau) d\tau = 1$. A nice property of mollifiers is their weak convergence to the Dirac distribution, that is

$$\lim_{\epsilon \to 0} \zeta(\tau) = \delta(\tau).$$

Let $[0, t_e]$ be the time interval, $t_0 = 0$ with $t_e > 0$[1]. Now, let t_κ be the time instant at which the pair (x_k, y_k) becomes available, let $t_\kappa < t_e$, and consider the functional

$$\mathcal{V}(w) = \int_0^{t_e} \psi(t) V(w(t)) dt$$

where

$$V(w(t)) = \sum_{\kappa=1}^{\ell} \zeta(t - t_\kappa) \cdot \overline{V}(f(x(t), w(t)), y(t))).$$

and $\psi \in C^\infty([0, t_e], \mathbb{R}^+)$ is a monotone increasing function which, in this paper, is chosen as $\psi(t) = e^{\beta t}$. As it will be shown later, this is related to energy dissipation and plays a crucial role for the establishment of the learning process. Basically, it prescribes a growing weight of the loss as time evolves. Now, let us assume $w \in \mathbb{R}^m$. We start to regard it as the *Lagrangian coordinates* of a virtual mechanical system. The learning problem defined by the supervised pairs $\mathcal{P}$, for a given choice of weights $w(t)$ at time t, defines a function $V(w(t))$ that, throughout this paper, is referred to as the *potential energy* of the system (neural network) defined by Lagrangian coordinates w. In machine learning, we look for trajectories $w(t)$ that possibly lead to configurations with small potential energy. The classic supervised learning is given a more adequate interpretation as $\epsilon \to 0$, which leads to replace the mollifiers with correspondent Dirac distributions $\delta(t - t_\kappa)$. When choosing the quadratic loss, we get

$$V(w(t)) = \sum_{\kappa=1}^{\ell} \delta(t - t_\kappa) \cdot (y(t) - f(x(t), w(t)))^2 .$$

The learning process in this case is only expected to reduce the error corresponding to the supervision points. For binary classification problems with $y(t) \in \{-1, 1\}$, however, if we adopt the hinge function

$$V(w(t)) = \sum_{\kappa=1}^{\ell} \delta(t - t_\kappa) \cdot \max\{\gamma - y(t) \cdot f(x(t), w(t)), 0\}$$

[1] It is of interest to consider also the case in which $t_e = \infty$.

being $\gamma > 0$ a proper threshold, we can promptly see that the learning process can led to the perfect match (zero loss) on some of the examples of the training set.

Now, following the duality with mechanics, we introduce the kinetic energy.

KINETIC ENERGY
Let $\mu_i > 0$ be the *mass* of each particle defined by the *position* $w_i(t)$ and *velocity* $\dot{w}_i$. Then, let us consider the *kinetic energy*

$$T(t) = \frac{1}{2} \sum_{i=1}^{m} \mu_i \dot{w}_i^2(t). \tag{1}$$

It gives a glimpse of the converge of the process of learning, since its end corresponds with $T = 0$. Like for the potential energy, in this paper we are interested in the accumulation $\int_0^{t_e} e^{\beta t} T(t) dt$ over $[0, t_e]$, which reflects the smoothness of the velocities of the particles. Moreover, also for the kinetic energy, we provide a growing account as time evolves which, as already stated, will be shown to be the basis of a dissipative behavior. The introduction of the exponential factor in both the potential and kinetic energy has been proposed in analytic mechanics as a way of introducing dissipation processes that are not present within the pure Hamiltonian framework [6].

VARIATIONAL FORMULATION OF LEARNING
Let us introduce the Lagrangian

$$\mathcal{L} := T - V$$

The problem of online learning can be formulated as that of finding

$$w^{\star} = \arg \min_{w \in \mathcal{W}} \int_0^{t_e} e^{\beta t} \mathcal{L}(w(t)) dt \tag{2}$$

3 Backprop from Euler-Lagrange Equations

The solution of the online learning problem can be obtained by finding stationary points of (2).

Theorem 1. *The solution of online learning stated by (2) satisfies*

$$\ddot{w}_i^{\star} + \beta \dot{w}_i^{\star} + \frac{1}{\mu_i} V'_{w_i} = 0, \tag{3}$$

where $V'_{w_i} = \sum_{\kappa=1}^{\ell} \overline{V}'_{w_i} \ \delta(t - t_\kappa)$.

Proof. We have

$$\frac{d}{dt}\frac{\partial}{\partial \dot{w}_i}\left(e^{\beta t}\mathcal{L}\right) = \frac{d}{dt}\frac{\partial}{\partial \dot{w}_i}\left(e^{\beta t} T\right) = \mu_i \frac{d}{dt}\left(e^{\beta t}\dot{w}_i\right) = \mu_i\left(e^{\beta t}\ddot{w}_i + \beta e^{\beta t}\dot{w}_i\right)$$

and

$$\frac{\partial}{\partial w_i}\left(e^{\beta t}\mathcal{L}\right) = -e^{\beta t}\frac{\partial V}{\partial w_i} = -e^{\beta t}V'_{w_i}.$$

Then the thesis follows when applying the Euler-Lagrange equation of (2). QED.

Notice that, since this theorem comes from the Euler-Lagrange equations, like for the action in analytic mechanics, the solution of the equations is not necessarily the absolute minimum. In general, it is a stationary point which is typically a saddle point. Interestingly, as shown in Section 4, like for other physical laws, this stationary point has nice minimization properties on the potential energy, which is exactly what we look for also in learning. Now let us assume that the system evolve from null Cauchy's conditions $w_i(0) = \dot{w}_i(0) = 0$ and let us use the notation $g_{i,\kappa} := \overline{V}'_{w_i}(w_i(t_\kappa))$. The following theorem holds true

Theorem 2. *The evolution from* null Cauchy's condition *follows the differential equation*

$$\frac{dw_i^\star}{dt} + \beta w_i^\star = -\frac{1}{\mu_i}\sum_{\kappa=1}^{\ell} g_{i,\kappa}\cdot 1(t-t_\kappa). \tag{4}$$

Proof. From Theorem 1 we have

$$\begin{aligned}\int_0^t \frac{d}{d\theta}\left(\frac{dw_i^\star}{d\theta} + \beta w_i^\star\right)\,d\theta &= -\frac{1}{\mu_i}\sum_{\kappa=1}^{\ell}\int_0^t \overline{V}'_{w_i}\cdot\delta(\theta-t_\kappa)\,d\theta\\ &= -\frac{1}{\mu_i}\sum_{\kappa=1}^{\ell} g_{i,\kappa}\cdot 1(t-t_\kappa).\end{aligned}$$

Now, the thesis follows when considering that $w_i^\star(0) = 0$ and $\dot{w}_i^\star(0) = 0$. QED.

Now, let us consider the answer to the first stimulus (supervised pair) coming at $t = t_1$ from the initial conditions $w_i(0) = \dot{w}_i(0) = 0$. We have

$$\frac{dw_i^\star}{dt} + \beta w_i^\star = -\frac{g_{i,1}}{\mu_i}.$$

If $w_i(0) = 0$ then

$$w_i^\star(t) = \frac{-g_{i,1}}{\beta\mu_i}\left(1 - e^{-\beta(t-t_1)}\right),$$

which indicates an asymptotic evolution to

$$\overline{w}_i^\star = \lim_{t\to\infty} w_i^\star(t) = \frac{-g_{i,1}}{\beta\mu_i}$$

Now we have $|w_i^\star(5/\beta) - \overline{w}_i^\star|/|\overline{w}_i^\star| < 0.01$, which means that with large values of β – or equivalently, small time constant $1/\beta$ – the weights are updated[2] from $w_i^\star(0) = w_i^\star|_0 = 0$ to $w_i^\star(1) = w_i^\star|_1$ by

$$w_i^\star(1) \simeq w_i^\star|_1 = -\eta_i\cdot g_{i,1} = -\frac{1}{\beta\mu_i}g_{i,1},$$

[2] We use the notation $w_i^\star|_t$ to indicate the corresponding *discrete updating* that are used in the on-line Backpropagation algorithm.

where $\eta_i := 1/(\beta\mu_i)$ is the classic *learning rate*. From now on, the notation $\simeq$ is used to indicate the above stated approximation of the asymptotic value $\overline{w}_i^\star$. Interestingly, the required high value for β corresponds with small learning rate, which is also kept small when considering particles with large mass μ_i. Beginning from this remark, now we establish the connection between the formulated continuous framework of learning with the classic on-line Backpropation algorithm.

Theorem 3. *Given $\mathcal{P} = \{x_\kappa, y_\kappa\}_{\kappa=1}^{\ell}$, where the supervised pairs (x_κ, y_κ) comes at $t = t_\kappa$, let us β such that $\forall \kappa = 1, \ldots, \ell$ we have*

$$\tau := 10/\beta \leq t_\kappa - t_{\kappa-1}. \tag{5}$$

Then

$$w_i^\star(t_k + \tau) \simeq w_i^\star(t_\kappa - \tau) - \eta_i g_{i,\kappa}, \tag{6}$$

which corresponds with the discrete counterpart

$$w_i^\star|_\kappa \simeq w_i^\star|_{\kappa-1} - \eta_i g_{i,\kappa}, \tag{7}$$

commonly referred to as the on-line Backpropagation algorithm.

Proof. We have

$$\int_{t_k-\tau}^{t_\kappa+\tau} \frac{d}{d\theta}\left(\frac{dw_i^\star}{d\theta} + \beta w_i^\star\right) d\theta = -\frac{1}{\mu_i}\sum_{\kappa=1}^{\ell}\int_{t_k-\tau}^{t_\kappa+\tau} \overline{V}'_{w_i}(w(t)) \cdot \delta(\theta - t_\kappa)d\theta,$$

from which we derive

$$\left(\frac{dw_i^\star}{d\theta} + \beta w_i^\star\right)_{t_\kappa+\tau} - \left(\frac{dw_i^\star}{d\theta} + \beta w_i^\star\right)_{t_\kappa-\tau} = -\frac{1}{\mu_i}\, 1(t - t_\kappa) g_{i,\kappa}.$$

Now, because of the strong damping hypothesis (5)

$$\left(\frac{dw_i^\star}{d\theta}\right)_{t_\kappa-\tau} \simeq 0 \;\; and \;\; w_i^\star(t_\kappa - \tau) \simeq w_i^\star|_{\kappa-1}$$

and, therefore, for $t > t_\kappa$ we get

$$\frac{dw_i^\star}{d\theta}|_{t_{\kappa+\tau}} + \beta w_i^\star|_{t_{\kappa+\tau}} - \beta w_i^\star|_{\kappa-1} \simeq -\frac{1}{\mu_i} g_{i,\kappa}.$$

Finally, the thesis follows when invoking again the strong damping hypothesis (5). QED.

4 Learning as a Dissipative Hamiltonian Process

Now we can establish a conservation principles that is related to dissipative systems[3]. From Theorem 1, if we multiply by $\dot{w}_i$ and accumulate over the weights, we get

$$\sum_{i=1}^{m} \mu_i\left(\dot{w}_i\ddot{w}_i + \beta\dot{w}_i^2\right) + \sum_{\kappa=1}^{\ell}\sum_{i=1}^{m} \overline{V}'_{w_i}(w_i(t))\; \dot{w}_i\; \delta(t - t_\kappa) = 0.$$

[3] For the sake of simplicity, in the following we drop the symbol $\star$.

Now we have

$$\frac{d\overline{V}(w(\theta))}{d\theta} = \sum_{i=1}^{m} \overline{V}'_{w_i}(w_i(t))\ \dot{w}_i.$$

If we accumulate over $[t_a, t_b]$ we get

$$\int_{t_a}^{t_b} \frac{d}{d\theta}\left(\frac{1}{2}\sum_{i=1}^{m}\mu_i \dot{w}_i^2\right) d\theta + \int_{t_a}^{t_b} \frac{d\overline{V}(w(\theta))}{d\theta} \cdot \sum_{\kappa=1}^{\ell} \delta(\theta - t_\kappa)\, d\theta + \int_{t_a}^{t_b} \beta \sum_{i=1}^{m} \mu_i \dot{w}_i^2\, d\theta = 0.$$

Now, if we define

$$D(t) := \int_0^t \beta \sum_{i=1}^{m} \mu_i \dot{w}_i^2\ d\theta = \frac{1}{\eta_i}\sum_{i=1}^{m}\int_0^t \dot{w}_i^2\ d\theta,$$

then, we get

$$\int_{t_a}^{t_b} \frac{d}{d\theta}\left(T + \overline{V}\sum_{\kappa=1}^{\ell}\delta(\theta - t_\kappa) + D\right) d\theta = 0.$$

Now, we use a notation overloading to denote by $T(t)$ the kinetic energy at t and we assume that $q \leq \ell$ supervised examples have been presented in $[0, t]$, being $t \in [t_a, t_b]$. If $\exists \kappa = 1, \ldots, \ell :\ t_\kappa \in [t_a, t_b]$ the above equation turns into the conservation equation

$$E(t) = T(t) + \overline{V}\sum_{\kappa=1}^{q} 1(t - t_\kappa) + D(t) = c \tag{8}$$

being c the constant energy of the extremes of the interval $[t_a, t_b]$. The overall energy $E(t)$ is conserved in all intervals in which there is no supervision. Whenever a supervised example is presented in the interval, the energy increases by

$$\overline{V}\sum_{\kappa=1}^{q}\left(1(t_\kappa) - 1(t_{\kappa-1})\right),$$

where $1(\cdot)$ is the Heaviside function. It turns out that the energy is injected by any supervised pairs, which yield new potential energy that is partly transformed into kinetic energy and partly dissipated. It is in fact the strong dissipation hypothesis given in terms of β which is responsible of producing stochastic gradient descent and which ensures the convergence of the learning process.

5 Conclusions

This paper gives a clean foundation of on-line Backpropagation in a variational framework by emphasizing strong connections with analytic mechanics. This approach can be thought of as the temporal counterpart of the study on regularization in the feature space given in [3]. It is shown that learning is in fact a dissipative process and that if the damping parameter β is large enough then we end up into the classic stochastic gradient descent scheme of on-line Backpropagation. This formulation might be of interest for exploring the new frontiers of lifelong learning models, in which we abandon learning processes based on training sets and consider intelligent agents living in their own environments. To this purpose, we have given natural laws expressed by second-order differential equations that obey an intriguing principle of energy conservation.

While all this quenches the desire of giving Backpropagation a formulation that resembles that of classic laws of Nature, the most attracting picture emerges when forcing small values of β, namely small dissipation in the learning process. In so doing, we depart significantly from stochastic gradient descent, and our preliminary connections with Statistical Mechanics indicate that if we learn with small dissipation we can gain more chance to get optimal solutions with respect to the traps that typically plague gradient descent. Intuitively, this is quite simple; the weights become particles whose behavior is that of a damping oscillation system, which is very well-suited to escape from local minima traps. Further research is needed to provide theoretical and experimental evidence of this intuition.

Acknowledgements. The research has been supported under the PRIN 2009 grant "Learning Techniques in Relational Domains and Their Applications".

References

1. Bottou, L., Bousquet, O.: The tradeoffs of large-scale learning. Advances in Neural Information Processing Systems 20, 161–168 (2008)
2. Bryson, A., Ho, Y.C.: Applied optimal control: optimization, estimation, and control. Blaisdell Publishing Company (1969)
3. Gnecco, G., Gori, M., Sanguineti, M.: Learning with boundary conditions. Neural computation 25(4), 1029–1106 (2013)
4. Gori, M., Maggini, M.: Optimal convergence of on-line backpropagation. IEEE Transactions on Neural Networks 7(1), 251–254 (1996)
5. Gori, M., Tesi, A.: On the problem of local minima in backpropagation. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(1), 76–86 (1992)
6. Herrera, L., Nunez, L., Patino, A., Rago, H.: A variational principle and the classical and quantum mechanics of the damped harmonic oscillator. American Journal of Physics 53(3), 273 (1985)
7. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Networks 4(2), 251–257 (1991)
8. Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
9. Werbos, P.J.: Prediction and analysis in the behavioral sciences. Tech. rep., Harvard University (1974)

GNMF with Newton-Based Methods

Rafał Zdunek[1], Anh-Huy Phan[2], and Andrzej Cichocki[2,3,4]

[1] Department of Electronics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
rafal.zdunek@pwr.wroc.pl
[2] Laboratory for Advanced Brain Signal Processing RIKEN BSI, Wako-shi, Japan
[3] Warsaw University of Technology, Poland
[4] Systems Research Institute, Polish Academy of Science (PAN), Poland

Abstract. Several variants of Nonnegative Matrix Factorization (NMF) have been proposed for supervised classification of various objects. Graph regularized NMF (GNMF) incorporates the information on the data geometric structure to the training process, which considerably improves the classification results. However, the multiplicative algorithms used for updating the underlying factors may result in a slow convergence of the training process. To tackle this problem, we propose to use the Spectral Projected Gradient (SPG) method that is based on quasi-Newton methods. The results are presented for image classification problems.

Keywords: NMF, Graph-regularized NMF, SPG, Image classification.

1 Introduction

Nonnegative Matrix Factorization (NMF) [1] decomposes a nonnegative matrix into lower-rank factor matrices that have nonnegative entries and usually some physical meaning. When NMF is applied to the matrix of training samples, we obtain sparse nonnegative feature vectors and coefficients of their nonnegative combination. The vectors of the coefficients lie in a low-dimensional latent component space. Hence, NMF is often regarded as a dimensionality reduction technique, and it has been widely applied for classification of various objects [2–6].

As reported in [7], the factor matrices obtained with NMF are generally nonunique. Several attempts have been done to additionally constrain them to satisfy a certain degree of sparsity, smoothness, uncorrelatedness, or orthogonality [2]. Cai *et al.* [8, 9] noticed that the projection from the high-dimensional observation space to the low-dimensional space should preserve the data geometrical structure. That is, any training samples forming one class should, after being projected, belong to the same class in the latent component space. Thus, they proposed Graph regularized NMF (GNMF) [8] that constrains one of the factor matrices with the information on the data geometric structure encoded in the nearest-neighbor graph of the training samples. This constraint was imposed to NMF by a specifically designed regularization term in the objective function

© Springer-Verlag Berlin Heidelberg 2013

that was then minimized with the standard multiplicative algorithm [2]. Guan *et al.* [10] considerably accelerated the convergence of GNMF by using additive gradient descent updates.

In this paper, we propose to improve the convergence rate of GNMF updates even more, by applying another Newton-based methods that provide the estimates according to the Karush-Kuhn-Tucker (KKT) optimality conditions. First, we formulate the Quadratic Programming (QP) problems for minimizing the penalized objective function. The QP problems can be efficiently solved with many numerical algorithms. To tackle large-scale classification problems, we suggest to use the modified Spectral Projected Gradient (SPG) method that belongs to the class of quasi-Newton methods. Moreover, we also propose to control the penalty parameters iteratively by some schedule included in the alternating update scheme.

The paper is organized in the following way. The next section discusses the Graph-regularized NMF. Section 3 is concerned with the optimization algorithms. The numerical experiments for image classification problems are presented in Section 4. Finally, the conclusions are drawn in Section 5.

2 Graph-Regularized NMF

Let $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T] \in \mathbb{R}_+^{I \times T}$, where $\boldsymbol{y}_t \in \mathbb{R}_+^I$ is the t-th training sample. Applying NMF to $\boldsymbol{Y}$, we get $\boldsymbol{Y} \cong \boldsymbol{AX}$, where the columns of the matrix $\boldsymbol{A} \in \mathbb{R}_+^{I \times J}$ represent the feature vectors, and the columns of the matrix $\boldsymbol{X} \in \mathbb{R}_+^{J \times T}$ are encoding vectors.

In several variants of NMF, the objective function can be expressed by the quadratic function:

$$\Psi(\boldsymbol{A}, \boldsymbol{X}) = \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{AX}||_F^2 + \frac{\alpha_X}{2}\operatorname{tr}(\boldsymbol{X}\boldsymbol{L}_X\boldsymbol{X}^T) + \frac{\alpha_A}{2}\operatorname{tr}(\boldsymbol{A}^T\boldsymbol{L}_A\boldsymbol{A}), \tag{1}$$

where $\boldsymbol{L}_X \in \mathbb{R}^{T \times T}$ and $\boldsymbol{L}_A \in \mathbb{R}^{I \times I}$ are symmetric weighting matrices. In supervised classification, the matrix $\boldsymbol{L}_X$ contains the information on assignments of the training samples to their classes. In DNMF [4], it is determined by the matrix of inner- and outer-class scattering. In GNMF [8], $\boldsymbol{L}_X$ is the graph Laplacian matrix that represents a data geometrical structure in the observation space. It takes form: $\boldsymbol{L}_X = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{W} = [w_{nm}] \in \mathbb{R}_+^{T \times T}$ contains the entries that determine the edges in the nearest neighbor graph of the observed points, and $\boldsymbol{D} = \operatorname{diag}\left(\sum_{m=1}^T w_{nm}\right) \in \mathbb{R}_+^{T \times T}$. The edges can be determined by the hard connections:

$$w_{nm} = \begin{cases} 1, \text{ if} & \boldsymbol{y}_n \in \mathcal{N}_p(\boldsymbol{y}_m), \text{ or } \boldsymbol{y}_m \in \mathcal{N}_p(\boldsymbol{y}_n), \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathcal{N}_p(\boldsymbol{y}_t)$ is the p nearest neighbor of the sample $\boldsymbol{y}_t$. We can also use the Heat kernel weighting:

$$w_{nm} = \begin{cases} \exp\left\{-\frac{||\boldsymbol{y}_n - \boldsymbol{y}_m||_2^2}{\sigma}\right\}, \text{ if} & \boldsymbol{y}_n \in \mathcal{N}_p(\boldsymbol{y}_m), \text{ or } \boldsymbol{y}_m \in \mathcal{N}_p(\boldsymbol{y}_n), \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

or the cosine measure:

$$w_{nm} = \begin{cases} \boldsymbol{y}_n^T \boldsymbol{y}_m, \text{ if } & \boldsymbol{y}_n \in \mathcal{N}_p(\boldsymbol{y}_m), \text{ or } \boldsymbol{y}_m \in \mathcal{N}_p(\boldsymbol{y}_n), \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The matrix $\boldsymbol{L}_A$ in (1) can enforce the smoothness in the feature vectors (the column vectors in $\boldsymbol{A}$) or other modality. We assumed the simplest approach to the smoothness by setting $\boldsymbol{L}_A = \boldsymbol{I}_I$, where $\boldsymbol{I}_I \in \mathbb{R}_+^{I \times I}$ is an identity matrix.

3 Algorithm

Since the matrix $\boldsymbol{L}_X$ in (1) is a symmetric and positive definite, the regularization term $\mathrm{tr}(\boldsymbol{X}\boldsymbol{L}_X\boldsymbol{X}^T)$ can be reformulated as follows:

$$\Psi_r(\boldsymbol{X}) = \mathrm{tr}(\boldsymbol{X}\boldsymbol{L}_X\boldsymbol{X}^T) = ||\boldsymbol{X}\boldsymbol{L}_X^{\frac{1}{2}}||_F^2 = ||(\boldsymbol{L}_X^{\frac{1}{2}} \otimes \boldsymbol{I}_J)\boldsymbol{x}||_2^2 = \boldsymbol{x}^T(\boldsymbol{L}_X \otimes \boldsymbol{I}_J)\boldsymbol{x}, \tag{5}$$

where $\boldsymbol{x} = \mathrm{vec}(\boldsymbol{X}) \in \mathbb{R}^{JT}$ is a vectorized form of $\boldsymbol{X}$, and $\otimes$ stands for the Kronecker product.

Considering the function (5), the minimization problem: $\min_{\mathbf{X}} \Psi(\boldsymbol{A}, \boldsymbol{X})$, s.t. $\boldsymbol{X} \geq \mathbf{0}$ can be expressed in terms of the Quadratic Programming (QP) problem: $\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}_X\boldsymbol{x} + \boldsymbol{c}_X^T\boldsymbol{x}, \quad \text{s.t.} \quad \boldsymbol{x} \geq 0$, where $\boldsymbol{Q}_X = \boldsymbol{I}_T \otimes \boldsymbol{A}^T\boldsymbol{A} + \alpha_X\boldsymbol{L}_X \otimes \boldsymbol{I}_J \in \mathbb{R}^{JT \times JT}$ and $\boldsymbol{c}_X = -\mathrm{vec}(\boldsymbol{A}^T\boldsymbol{Y}) \in \mathbb{R}^{JT}$.

Similarly, the matrix $\boldsymbol{A}$ can be also computed by formulating the QP problem: $\min_{\boldsymbol{a}} \frac{1}{2}\boldsymbol{a}^T\boldsymbol{Q}_A\boldsymbol{a} + \boldsymbol{c}_A^T\boldsymbol{a}, \quad \text{s.t.} \quad \boldsymbol{a} \geq 0$, where $\boldsymbol{a} = \mathrm{vec}(\boldsymbol{A}^T) \in \mathbb{R}^{IJ}$, $\boldsymbol{Q}_A = (\boldsymbol{X}\boldsymbol{X}^T + \alpha_A\boldsymbol{I}_J) \otimes \boldsymbol{I}_I \in \mathbb{R}^{IJ \times IJ}$ and $\boldsymbol{c}_A = -\mathrm{vec}(\boldsymbol{Y}\boldsymbol{X}^T) \in \mathbb{R}^{IJ}$.

Since the function (1) is quadratic with respect to both arguments $\boldsymbol{A}$ and $\boldsymbol{X}$ (but not jointly), the matrices $\boldsymbol{Q}_A$ and $\boldsymbol{Q}_X$ are equivalent to the Hessian matrices for $\boldsymbol{A}$ and $\boldsymbol{X}$, respectively. When $\alpha_A > 0$, the matrix $\boldsymbol{Q}_A$ is positive definite. Under the assumption of positive definiteness of the matrix $\boldsymbol{L}_X$, the matrix $\boldsymbol{Q}_X$ is also positive definite. Hence, both QP problems are strictly convex. To solve such problems, we can use many numerical algorithms such as the Active-Set (AS), Interior-Point (IP), and Spectral Projected Gradient (SPG) [11]. These algorithm are based on the Newton or quasi-Newton updates.

Note that the matrix $\boldsymbol{Q}_A$ has a block-diagonal structure, and hence the updates of $\boldsymbol{A}$ might be considerably accelerated by transforming the nonnegative least-squares problem: $\min_{\mathbf{A} \geq 0} \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}||_F^2 + \frac{\alpha_A}{2}||\boldsymbol{A}||_F^2$ to the normal equations $\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A}^T = \boldsymbol{X}\boldsymbol{Y}^T$ subject to the nonnegativity constraints $\boldsymbol{A} \geq \mathbf{0}$. Then, the solution can be efficiently searched with the FC-NNLS algorithm that was proposed by Benthem and Keenan [12], and then adapted to NMF problems in [13].

The updates for $\boldsymbol{X}$ cannot be accelerated in the similar way, however, there is still a possibility of applying some quasi-Newton method without formulating the Hessian $\boldsymbol{Q}_X$. Note that the matrix $\boldsymbol{Q}_X$ is very large when the number of training samples is large, and it is rather a dense matrix due to the matrix $\boldsymbol{L}_X$. One of these possibilities is to use the SPG method [14] that combines the standard gradient projection scheme with the nonmonotonic Barzilai-Borwein

(BB) method [11]. It is used for minimization of convex functions subject to box-constraints.

In the SPG method, the descent direction $\boldsymbol{p}_t^{(k)}$ for updating the vector $\boldsymbol{x}_t$ in the k-th iteration is defined as follows:

$$\boldsymbol{p}_t^{(k)} = \left[\boldsymbol{x}_t^{(k)} - (\alpha_t^{(k)})^{-1}\nabla_{\boldsymbol{x}_t}\Psi(\boldsymbol{A}, \boldsymbol{x}_t^{(k)})\right]_+ - \boldsymbol{x}_t^{(k)}, \tag{6}$$

for $\alpha_t^{(k)} > 0$ selected in such a way that the matrix $\alpha_t^{(k)}\boldsymbol{I}_J$ approximates the Hessian matrix.

In [2], this method was adopted to parallel processing of all column vectors in $\boldsymbol{X}$. Using this approach, we have the update rule:

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + \boldsymbol{P}^{(k)}\boldsymbol{Z}^{(k)}, \tag{7}$$

where $\boldsymbol{Z}^{(k)} = \operatorname{diag}\{\boldsymbol{\eta}^{(k)}\}$. The column vectors of $\boldsymbol{P}^{(k)} \in \mathbb{R}^{J\times T}$ and the entries of the vector $\boldsymbol{\eta}^{(k)} \in \mathbb{R}_+^T$ are descent directions and steplengths for updating the vectors $\{\boldsymbol{x}_t\}$, respectively. According to (6), the matrix $\boldsymbol{P}^{(k)}$ has the form:

$$\boldsymbol{P}^{(k)} = \left[\boldsymbol{X}^{(k)} - \boldsymbol{G}_X^{(k)}\boldsymbol{D}^{(k)}\right]_+ - \boldsymbol{X}^{(k)}, \tag{8}$$

where $\boldsymbol{G}_X^{(k)} = \nabla_{\mathbf{X}}\Psi(\boldsymbol{A}, \boldsymbol{X}^{(k)}) \in \mathbb{R}^{J\times T}$ and $\boldsymbol{D}^{(k)} = \operatorname{diag}\{(\alpha_t^{(k)})^{-1}\} \in \mathbb{R}^{T\times T}$.

The coefficients $\{\alpha_t^{(k)}\}$ can be obtained from the secant equation that is given by $\boldsymbol{S}^{(k)}\operatorname{diag}\{\alpha_t^{(k+1)}\} = \boldsymbol{W}^{(k)}$, where $\boldsymbol{S}^{(k)} = \boldsymbol{X}^{(k+1)} - \boldsymbol{X}^{(k)}$ and $\boldsymbol{W}^{(k)} = \nabla_{\mathbf{X}}\Psi(\boldsymbol{A}, \boldsymbol{X}^{(k+1)}) - \nabla_{\mathbf{X}}\Psi(\boldsymbol{X}^{(k)})$. For the minimization of the objective function (1) with respect to $\boldsymbol{X}$, the matrix $\boldsymbol{W}^{(k)}$ takes the form: $\boldsymbol{W}^{(k)} = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{S}^{(k)} + \alpha_X\boldsymbol{S}^{(k)}\boldsymbol{L}_X$. From (7) we have: $\boldsymbol{S}^{(k)} = \boldsymbol{P}^{(k)}\boldsymbol{Z}^{(k)}$. In consequence, the secant equation leads to:

$$\begin{aligned}\boldsymbol{\alpha}^{(k+1)} &= \frac{\operatorname{diag}\left\{(\boldsymbol{S}^{(k)})^T\boldsymbol{W}^{(k)}\right\}}{\operatorname{diag}\left\{(\boldsymbol{S}^{(k)})^T\boldsymbol{S}^{(k)}\right\}} = \frac{\operatorname{diag}\left\{(\boldsymbol{S}^{(k)})^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{S}^{(k)} + \alpha_X(\boldsymbol{S}^{(k)})^T\boldsymbol{S}^{(k)}\boldsymbol{L}_X\right\}}{\operatorname{diag}\left\{(\boldsymbol{S}^{(k)})^T\boldsymbol{S}^{(k)}\right\}} \\ &= \frac{\operatorname{diag}\left\{(\boldsymbol{P}^{(k)})^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{P}^{(k)} + \alpha_X(\boldsymbol{P}^{(k)})^T\boldsymbol{P}^{(k)}\boldsymbol{Z}^{(k)}\boldsymbol{L}_X(\boldsymbol{Z}^{(k)})^{-1}\right\}}{\operatorname{diag}\left\{(\boldsymbol{P}^{(k)})^T\boldsymbol{P}^{(k)}\right\}} \\ &= \frac{\mathbf{1}_J^T\left[\boldsymbol{P}^{(k)} \circledast \left(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{P}^{(k)} + \alpha_X\boldsymbol{P}^{(k)}\boldsymbol{Z}^{(k)}\boldsymbol{L}_X(\boldsymbol{Z}^{(k)})^{-1}\right)\right]}{\mathbf{1}_J^T\left[\boldsymbol{P}^{(k)} \circledast \boldsymbol{P}^{(k)}\right]},\end{aligned} \tag{9}$$

where $\circledast$ stands for the Hadamard product, and the operation $\operatorname{diag}\{\boldsymbol{M}\}$ creates a vector containing the main diagonal entries of a matrix $\boldsymbol{M}$. Note that the matrix $\boldsymbol{Z}^{(k)}$ is diagonal, so the product $\boldsymbol{Z}^{(k)}\boldsymbol{L}_X(\boldsymbol{Z}^{(k)})^{-1}$ can be readily calculated.

The steplengths can be estimated by solving the minimization problem:

$$\boldsymbol{\eta}_*^{(k)} = \arg\min_{\boldsymbol{\eta}^{(k)}} \Psi\left(\boldsymbol{A}, \boldsymbol{X}^{(k)} + \boldsymbol{P}^{(k)}\operatorname{diag}\{\boldsymbol{\eta}^{(k)}\}\right). \tag{10}$$

If $\alpha_X = 0$, the problem (10) can be expressed in a closed-form. Otherwise, iterative updates must be used, e.g. the Armijo rule [11].

The final form of the modified SPG algorithm is given by Algorithm 1. The final form of the NMF algorithm used in the training process is given by Algorithm 2.

Algorithm 1. SPG algorithm

```
Input : $\boldsymbol{Y} \in \mathbb{R}_+^{I\times T}$, $\boldsymbol{A} \in \mathbb{R}_+^{I\times J}$, $\boldsymbol{X}^{(0)} \in \mathbb{R}_+^{J\times T}$ - initial guess, $k_{max}$ - number of
        iterations for SPG updates, $\alpha_{min} > 0$, $\alpha_{max} > 0$, $\forall t : \bar{\alpha}_t^{(0)} = \frac{1}{2}\alpha_{max}$,
Output: $\hat{\boldsymbol{X}}$ - estimated factor matrices,
1 for $k = 0, 1, \ldots, k_{max}$ do
2     $\boldsymbol{G}_X^{(k)} = \nabla_{\mathbf{X}} \Psi(\boldsymbol{A}, \boldsymbol{X}^{(k)}) = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{X}^{(k)} - \boldsymbol{Y}) + \alpha_X \boldsymbol{X}^{(k)} \boldsymbol{L}_X$ ;   // Gradient
3     $\boldsymbol{P}^{(k)} = \left[\boldsymbol{X}^{(k)} - \boldsymbol{G}_X^{(k)} \operatorname{diag}\{(\bar{\alpha}_t^{(k)})^{-1}\}\right]_+ - \boldsymbol{X}^{(k)}$ ;   // Descent direction
4     $\bar{\boldsymbol{\eta}}^{(k)} = \max\{0, \min\{1, \boldsymbol{\eta}^{(k)}\}\}$ ;   // where $\boldsymbol{\eta}^{(k)}$ is estimated with (10)
5     $\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + \boldsymbol{P}^{(k)} \operatorname{diag}\{\bar{\boldsymbol{\eta}}^{(k)}\}$;
6     $\bar{\boldsymbol{\alpha}}^{(k+1)} = \max\{\alpha_{min}, \min\{\alpha_{max}, \boldsymbol{\alpha}^{(k+1)}\}\}$; // where $\boldsymbol{\alpha}^{(k+1)}$ is set to (9)
```

In the training process, we obtain the nonnegative matrices $\boldsymbol{A}$ and $\boldsymbol{X}$. The column vectors of $\boldsymbol{X}$ contain the discriminant information. To classify the test sample $\tilde{\boldsymbol{y}}$, first we need to project it onto the subspace spanned by the column vectors of the matrix $\boldsymbol{A}$. As a result, we obtain $\tilde{\boldsymbol{x}} \in \mathbb{R}_+^J$. This step can be carried out with the SPG, assuming $\alpha_X = 0$. Then, the following problem is solved: $t_* = \arg\min_{1\leq t\leq T} ||\tilde{\boldsymbol{x}} - \boldsymbol{x}_t||_2$, which gives the index t_* of the class to which the sample $\tilde{\boldsymbol{y}}$ is classified.

4 Experiments

The experiments are carried out for classification of facial images taken from the ORL database[1]. It contains 400 frontal facial images of 40 people (10 pictures per person). We selected 8 training images randomly from each class, and the remaining 2 images are used for testing.

We test the following NMF algorithms: MUE (standard multiplicative Lee-Seung algorithm for the Euclidean distance) [1], GNMF [8], MD-NMF [10], standard projected ALS [2], LPG (Lin's Projected Gradient) [15], IP (Interior-Point NMF) [16], regularized FC-NNLS [13], SPG-NMF (Algorithm 2). For the SPG algorithm, we found the optimal parameters: $\alpha_X = 10^{-5}$, $\bar{\alpha} = 10^{-12}$, $\alpha_0 = 0.01$, and $k_{max} = \min\{k, 50\}$, where k is the alternating step in Algorithm 2. The matrix $\boldsymbol{L}_X$ is determined using the hard connection criterion given by (2). The iterative process is terminated after 50 alternating steps.

[1] http://people.cs.uchicago.edu/~dinoj/vis/orl/

Algorithm 2. SPG-NMF Algorithm

```
   Input  : $\boldsymbol{Y} \in \mathbb{R}^{I \times T}$, $J$ - lower rank, $\alpha_0$ - initial regularization parameter,
   Output: Factor matrices: $\boldsymbol{A} \in \mathbb{R}_+^{I \times J}$ and $\boldsymbol{X} \in \mathbb{R}_+^{J \times T}$
1  Initialize: $\boldsymbol{A}$ and $\boldsymbol{X}$ with nonnegative random numbers;
2  Replace negative entries (if any) in $\boldsymbol{Y}$ with zero-value, $k = 0$ ;
3  repeat
4      $\alpha_A^{(k)} = \max\left\{\bar{\alpha}, 2^{-k}\alpha_0\right\}$ ;          // Regularization parameter schedule
5      $\boldsymbol{X}^{(k+1)} = \texttt{SPG}(\boldsymbol{Y}, \boldsymbol{A}^{(k)}, \boldsymbol{X}^{(k)}, \alpha_X)$;
6      $\bar{d}_j^{(k+1)} = \sum_{t=1}^{T} x_{jt}^{(k+1)}$,
       $\boldsymbol{X}^{(k+1)} \leftarrow \text{diag}\left\{\left(\bar{d}_j^{(k+1)}\right)^{-1}\right\}\boldsymbol{X}^{(k+1)}$,   $\boldsymbol{A}^{(k)} \leftarrow \boldsymbol{A}^{(k)}\text{diag}\left\{\bar{d}_j^{(k+1)}\right\}$;
7      $\bar{\boldsymbol{A}}^{(k+1)} = \texttt{FCNNLS}(\boldsymbol{Y}^T, (\boldsymbol{X}^{(k+1)})^T, (\boldsymbol{A}^{(k)})^T, \alpha_A^{(k)})$;
8      $\boldsymbol{A}^{(k+1)} = (\bar{\boldsymbol{A}}^{(k+1)})^T$;
9      $\bar{\bar{d}}_j^{(k+1)} = \sum_{i=1}^{I} a_{ij}^{(k+1)}$,
       $\boldsymbol{X}^{(k+1)} \leftarrow \text{diag}\left\{\bar{\bar{d}}_j^{(k+1)}\right\}\boldsymbol{X}^{(k+1)}$,   $\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k+1)}\text{diag}\left\{\left(\bar{\bar{d}}_j^{(k+1)}\right)^{-1}\right\}$;
10     $k \leftarrow k + 1$;
11 until Stop criterion is satisfied;
```

The NMF algorithms are initialized with uniformly distributed random matrices, and tested for various values of the related parameters. Fig. 1 presents the mean recognition rate versus the number of components (parameter J) obtained with different NMF algorithms.

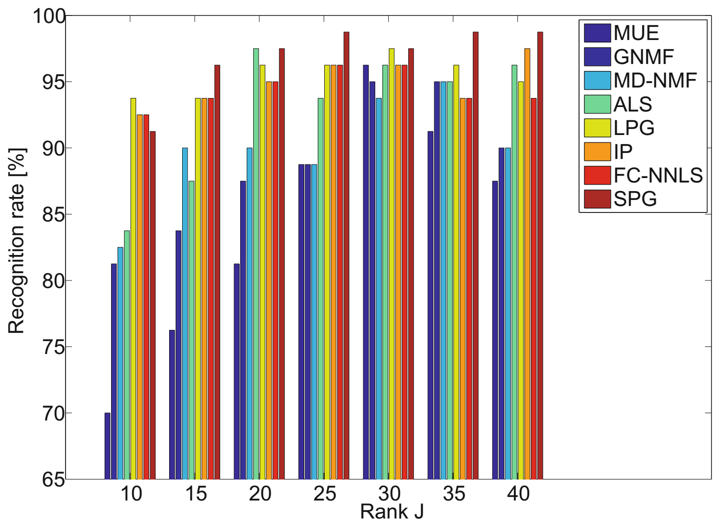


Fig. 1. Recognition rate obtained using various NMF algorithms versus the number of components J

The normalized residual errors versus the number of iterations for the selected NMF algorithms are plotted in Fig. 2.

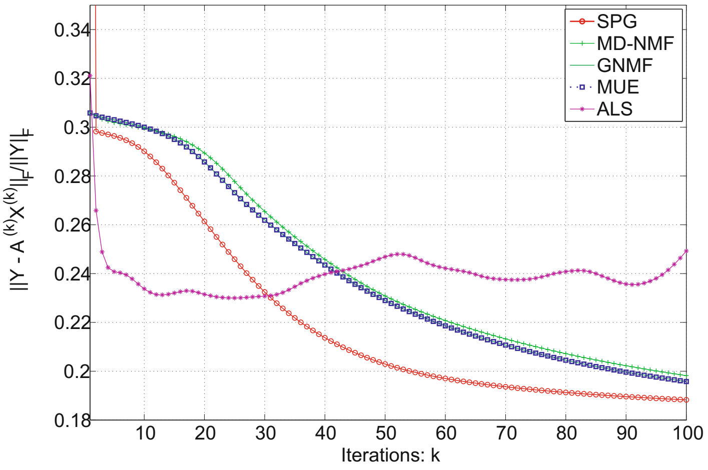


Fig. 2. Normalized residual errors versus alternating iterations

5 Conclusions

The results presented in Fig. 1 demonstrate that the SPG-NMF algorithm outperforms the other tested algorithms in terms of the recognition rate for $J > 10$. Usually an increase in the factorization rank leads to a higher recognition rate. The experiments also confirm that the NMF algorithms based on Newton-like methods (SPG, IP, FC-NNLS and ALS) converge faster than the multiplicative algorithms. This can be observed in Fig. 2 where the SPG-NMF algorithm demonstrates a better convergence behavior than the others. Initially the projected ALS algorithm converges faster but it does not guarantee a monotonic convergence. As observed in Fig. 2 the residual error of the SPG-NMF decreases monotonically with alternating steps. This behavior is also justified by the fact that both SPG and FC-NNLS algorithms converge to the solution optimal according to the KKT conditions. Moreover, the convergence of the SPG-NMF is faster than for the multiplicative algorithms since the SPG is a quasi-Newton method, i.e., the gradient direction is scaled using the information on the Hessian approximation.

Summing up, the experiments showed that the proposed algorithm works very efficiently for the facial classification problem. The usefulness of the proposed algorithm in other applications of NMF will be analyzed in the further research.

References

1. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)

2. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.I.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley and Sons (2009)
3. Qin, L., Zheng, Q., Jiang, S., Huang, Q., Gao, W.: Unsupervised texture classification: Automatically discover and classify texture patterns. Image and Vision Computing 26(5), 647–656 (2008)
4. Zafeiriou, S., Tefas, A., Buciu, I., Pitas, I.: Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. IEEE Transactions on Neural Networks 17(3), 683–695 (2006)
5. Guillamet, D., Vitria, J.: Classifying faces with nonnegative matrix factorization. In: Proc. 5th Catalan Conference for Artificial Intelligence, Castello de la Plana, Spain, pp. 24–31 (2002)
6. Benetos, E., Kotti, M., Kotropoulos, C.: Musical instrument classification using non-negative matrix factorization algorithms and subset feature selection. In: Proc. of 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006), Toulouse, France (2006)
7. Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 16. MIT Press, Cambridge (2004)
8. Cai, D., He, X., Wu, X., Han, J.: Nonnegative matrix factorization on manifold. In: Proc. 8-th IEEE International Conference on Data Mining (ICDM), pp. 63–72 (2008)
9. Cai, D., He, X., Han, J., Huang, T.: Graph regularized nonnegative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(8), 1548–1560 (2011)
10. Guan, N., Tao, D., Luo, Z., Yuan, B.: Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent. IEEE Transactions on Image Processing 20(7), 2030–2048 (2011)
11. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research. Springer, New York (1999)
12. Benthem, M.H.V., Keenan, M.R.: Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. Journal of Chemometrics 18, 441–450 (2004)
13. Kim, H., Park, H.: Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. SIAM Journal in Matrix Analysis and Applications 30(2), 713–730 (2008)
14. Birgin, E.G., Martnez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Control and Optimization 10, 1196–1211 (2000)
15. Lin, C.J.: Projected gradient methods for non-negative matrix factorization. Neural Computation 19(10), 2756–2779 (2007)
16. Zdunek, R.: Spectral signal unmixing with interior-point nonnegative matrix factorization. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 65–72. Springer, Heidelberg (2012)

Improving the Associative Rule Chaining Architecture

Nathan Burles, Simon O'Keefe, and James Austin

Advanced Computer Architectures Group,
Department of Computer Science,
University of York,
York, YO10 5GH, UK
{nburles,sok,austin}@cs.york.ac.uk
http://www.cs.york.ac.uk

Abstract. This paper describes improvements to the rule chaining architecture presented in [1]. The architecture uses distributed associative memories to allow the system to utilise memory efficiently, and superimposed distributed representations in order to reduce the time complexity of a tree search to $O(d)$, where d is the depth of the tree. This new work reduces the memory required by the architecture, and can also further reduce the time complexity.

Keywords: rule chaining, correlation matrix memory, associative memory, distributed representation, parallel distributed computation.

1 Introduction

Rule chaining is a common problem in the field of artificial intelligence; searching a tree of rules to determine if there is a path from the starting state to the goal state. The Associative Rule Chaining Architecture (ARCA) [1] uses correlation matrix memories (CMMs)—a simple associative neural network [2]—to perform rule chaining. We present an improvement to the original ARCA architecture that reduces the memory required, and can also reduce the time complexity.

Rule chaining includes both forward and backward chaining. In this work we describe the use of forward chaining, working from the starting state towards the goal state, although there is no reason that backward chaining could not be used with this architecture.

In forward chaining, the search begins with an initial set of conditions that are known to be true. Each of the rules is then checked in turn, to find one for which the antecedents match these conditions. The consequents of that rule are then added to the current state, which is checked to decide if the goal has been found. If it has not, then the search continues by iterating—if no further rules are found to match then the search results in failure.

This is essentially a tree search, and so classical algorithms such as depth-first search are commonly used. The time complexity of such an algorithm is $O(b^d)$, where b is the branching factor and d is the depth of the tree. Reducing this to $O(d)$ therefore represents a potentially significant improvement.

© Springer-Verlag Berlin Heidelberg 2013

1.1 Correlation Matrix Memories (CMMs)

The CMM is a simple, fully connected, associative neural network consisting of a single layer of weights. Despite their simplicity, associative networks are still an active area of research (e.g. [3,4]). In this work we use a sub-class of CMMs, where these weights are restricted to binary values, known as binary CMMs [5].

Binary CMMs use simple Hebbian learning [6]. Learning to associate pairs of binary vectors is thus an efficient operation, requiring only local updates to the CMM. This learning is formalised in Equation 1, where $\mathbf{M}$ is the resulting CMM (matrix of binary weights), $\mathbf{x}$ is the set of input vectors, $\mathbf{y}$ is the set of output vectors, n is the number of training pairs, and $\bigvee$ indicates the logical OR of binary vectors or matrices.

$$\mathbf{M} = \textstyle\bigvee_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i^T \tag{1}$$

A recall operation may be performed as shown in Equation 2. A matrix multiplication between the transposed input vector and the CMM results in a non-binary output vector, to which a threshold function f must be applied in order to produce the final output vector.

$$\mathbf{y} = f(\mathbf{x}^T \mathbf{M}) \tag{2}$$

It is possible to greatly optimise this recall operation, using the fact that the input vector contains only binary components. For the j^{th} bit in the output, the result of a matrix multiplication is the vector dot product of the transposed vector $\mathbf{x}^T$ and the j^{th} column of matrix $\mathbf{M}$. In turn the vector dot product is defined as $\sum_{i=1}^{n} \mathbf{x}_i \mathbf{M}_{j,i}$, where $\mathbf{M}_{j,i}$ is the value stored in the j^{th} column of the i^{th} row of the CMM $\mathbf{M}$. Given the binary nature of $\mathbf{x}$ it is clear that this dot product is equal to the sum of all values $\mathbf{M}_{j,i}$ where $\mathbf{x}_i = 1$, formalised in Equation 3.

$$\mathbf{y}_j = f(\textstyle\sum_{i(\mathbf{x}_i=1)} \mathbf{M}_{j,i}) \tag{3}$$

There are various options as to which threshold function, f, may be applied during recall. The choice of function depends on the application, and on the data representation used. ARCA uses superposition of vectors, so the selection is limited to Willshaw thresholding, where any output bit with a value at least equal to the (fixed) trained input weight is set to one [5].

1.2 Associative Rule Chaining

The Associative Rule Chaining Architecture stores multiple states superimposed in a single vector using a distributed representation [7], which also helps to provide more efficient memory use and a greater tolerance to faults than a local representation [8]. For example, the superposition of two vectors {`0 0 1 0 0`} and {`1 0 0 0 0`} is the vector {`1 0 1 0 0`}.

ARCA performs rule chaining using superimposed representations, hence reducing the time complexity of a tree search. The main challenge overcome by the architecture is to maintain the separation of each superimposed state throughout the search, without needing to separate out the distributed patterns or revert to a local representation.

To solve this challenge, each rule is assigned a unique "rule vector" which exists in a separate vector space to those used for the antecedent and consequent tokens. ARCA stores the antecedents and consequents of rules in two separate CMMs, connected by the rule vector [1], described further in Section 2.

2 Improving the ARCA Architecture

In the original architecture, two CMMs are used to separate the antecedents and consequents of rules. When storing a rule, for example $a \rightarrow b$, a unique "rule vector" must be generated. This is essentially a label for the rule, but can be considered as our example rule becoming $a \rightarrow r_0 \rightarrow b$.

The first CMM is used to store the associations between the superimposed antecedents of a rule (a) and the assigned rule vector (r_0). This results in the rule firing if the tokens in the head of each rule are contained within a presented input, i.e. $a \rightarrow r_0$.

When training the second CMM, a slightly more complex method is required. Initially, a tensor product (TP) is formed between the superimposed consequents of a rule (b) and the rule vector (r_0); this TP is "flattened" in row-major order to form a vector ($b : r_0$). The associations between the rule vector and this TP are then stored in the second CMM. This means that when a rule fires from the antecedent CMM, the consequent CMM will produce a TP containing the output tokens bound to the rule that caused them to fire, i.e. $r_0 \rightarrow (b : r_0)$. These tokens can then be added to the current state in order to iterate.

2.1 Using a Single CMM

In ARCA the separation of superimposed states during the recall process is actually performed by the rule vector, rather than through the use of two CMMs. We propose that this is unnecessary and we can use a single CMM mapping directly from the antecedents to the consequents, reducing both the memory required to store the rules and the time required to perform a recall operation.

To train this reduced ARCA requires a similar operation as originally used when training the second CMM. Every rule is still assigned a unique rule vector, which is used to form a TP with the superimposed consequents of the rule ($b : r_0$). The single CMM is now trained using the superimposed antecedents of the rule as an input and this TP as an output, i.e. $a \rightarrow (b : r_0)$. Upon presentation of an input containing the tokens in the head of a rule, the CMM will produce a TP containing the output tokens bound to the rule that caused them to fire.

2.2 Recall

Fig. 1 shows a recall process performed on the reduced ARCA. To initialise the recall, an input state TP_{in} is created by forming the TP of any initial tokens

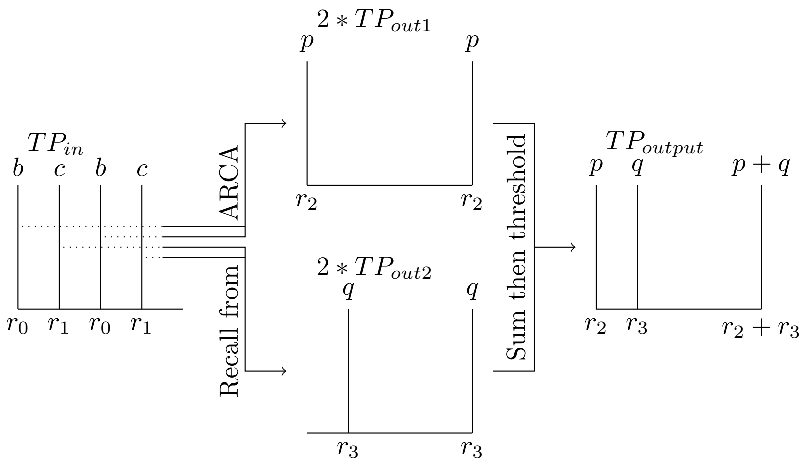


Fig. 1. A visualisation of the recall process within the reduced ARCA. The tensor products (TPs) contain different token vectors bound to rule vectors. Each column is labelled at the top with the tokens contained within the column. The position of each column is defined by the positions of the bits set to one in the rule vector to which the token is bound, labelled at the base of the column. The remainder of the TP consists solely of zeros. The vector weight is 2, and hence each column appears twice.

with a rule vector. In the diagram we assume that the vector weight is 2, and we initialise the search with two separate, but superimposed, inputs—$b : r_0$ and $c : r_1$.

The recall process can now begin, by recalling each column of TP_{in} from the CMM in turn. The result of each column recall is an entire TP of equal dimensions to the original input (TP_{outx}) containing the consequents of a rule, bound to the rule that fired them. In our diagram each of these output TPs is recalled twice, once for each column in TP_{in}.

We need to combine these to form a single TP to allow the system to iterate. As can be seen in the diagram, any antecedents will appear in TP_{in} a number of times equal to the weight of a rule vector. Thus, the consequents will appear in the same number of TP_{out}s, bound to the rule that caused them to fire. When these are summed, the weight of a rule vector can therefore be used as a threshold to obtain the final output—a single binary TP, TP_{output}.

Before the system iterates, we need to check whether the search has completed. Firstly we can check whether the search should continue. If TP_{output} consists solely of zeros, then no rules have been matched and hence the search is completed without finding a goal state.

If TP_{output} is not empty, then we must check whether a goal state has been reached. This is achieved by treating TP_{output} as a CMM. The superposition of the goal tokens is used as an input to this CMM, and the threshold set to the combined weight of these superimposed goal tokens. If the resulting binary vector contains a rule vector, then this indicates that this rule vector was bound to the goal token and we can conclude that the goal state has been reached.

2.3 Time Complexity of the Reduced ARCA

We have previously shown [1] that ARCA is able to search multiple branches of a tree in parallel, while maintaining separation between them. This reduces the time complexity of a search to $O(d)$, where d is the depth of the tree. Contrasted with a depth-first approach with a time complexity of $O(b^d)$, where b is the branching factor, this is a notable improvement.

In order to compare the worst case time complexity of the original and the reduced ARCA, we now perform a more detailed analysis. When binary matrices and vectors are stored sparsely, the time complexity of a simple CMM recall operation depends on the weight of the input vector $w_{\mathbf{x}}$ and the number of bits set in each row of the matrix $w_{\mathbf{M}[i]}$. In the worst case, $w_{\mathbf{M}[i]}$ will be equal to the length of the output vector $l_{\mathbf{y}}$.

Using Equation 3, applied to all columns, the time complexity is found as the time to sum each row $\mathbf{M}[i]$ where $\mathbf{x}_i = 1$, plus a linear scan through the output vector to apply the final threshold. This is given in Equation 4.

$$\mathrm{TC}_{\mathrm{recall}} = w_{\mathbf{x}} l_{\mathbf{y}} + l_{\mathbf{y}} \tag{4}$$

In applying Equation 4 to the ARCA, we must consider the lengths and weights of vectors used to represent tokens and rules as potentially different, resulting in four terms: $l_{\mathbf{t}}$, $w_{\mathbf{t}}$, $l_{\mathbf{r}}$, and $w_{\mathbf{r}}$—representing the lengths and weights of tokens and rules respectively.

It is also important to remember that this equation calculates the time complexity of the recall of a single vector, where in ARCA every column of an input TP is recalled in turn. As both the original and the reduced ARCA operate in the same fashion, this multiplier can be ignored for the purpose of this comparison.

The worst case time complexity of recalling a single vector from the original ARCA and the reduced ARCA can now be derived to find Equations 5 and 6 respectively.

$$\begin{aligned} \mathrm{TC}_{\mathrm{original}} &= \mathrm{CMM1} + \mathrm{CMM2} \\ &= (w_{\mathbf{t}} l_{\mathbf{r}} + l_{\mathbf{r}}) + (w_{\mathbf{r}} l_{\mathbf{t}} l_{\mathbf{r}} + l_{\mathbf{t}} l_{\mathbf{r}}) \\ &= l_{\mathbf{r}}(w_{\mathbf{t}} + 1 + l_{\mathbf{t}}(w_{\mathbf{r}} + 1)) \end{aligned} \tag{5}$$

$$\begin{aligned} \mathrm{TC}_{\mathrm{reduced}} &= w_{\mathbf{t}} l_{\mathbf{t}} l_{\mathbf{r}} + l_{\mathbf{t}} l_{\mathbf{r}} \\ &= l_{\mathbf{r}} l_{\mathbf{t}}(w_{\mathbf{t}} + 1) \end{aligned} \tag{6}$$

In order to find the relative parameters for which the reduced ARCA becomes more efficient than the original ARCA we equate 5 and 6 and simplify as far as possible, as in Equation 7.

$$\begin{aligned} l_{\mathbf{r}}(w_{\mathbf{t}} + 1 + l_{\mathbf{t}}(w_{\mathbf{r}} + 1)) &= l_{\mathbf{r}} l_{\mathbf{t}}(w_{\mathbf{t}} + 1) \\ 1 + l_{\mathbf{t}} w_{\mathbf{r}} &= w_{\mathbf{t}}(l_{\mathbf{t}} - 1) \end{aligned} \tag{7}$$

This equation shows that in the worst case, if the weight of both rule and token vectors is equal, that the original and the reduced ARCA will perform essentially equally. If the weight of rule vectors is greater than that of token vectors, then the reduced ARCA will outperform the original.

In reality, the worst case is not possible—if all of the rows of the matrix were completely full, then it would be impossible to successfully recall any vector that was originally stored. Rather than attempting to time a comparison, as this is unlikely to provide accurate or fair results, we instead compare the memory requirements experimentally.

2.4 Comparison of Memory Requirements

The time complexity of a recall operation is dependent on the number of bits set in each row of the matrix. As such, a comparison of the memory required by the original and the reduced ARCA also provides a good indication of any improvement in the time complexity.

In order to compare the memory required, both variants of ARCA have been applied to the same randomly generated problems. For each experiment a tree of rules was generated with a given depth d and maximum branching factor b, using the same procedure as detailed in previous work [1]. These rules were then learned by both systems, and rule chaining was performed on them.

In all experiments, we fixed the vector weight for both rules and tokens to be 4. This value results in sparse vectors over the range of vector lengths investigated, and should provide good performance in the CMMs [9].

Given our analysis of the time complexity, using the same weight in both types of vector would also be expected to result in very similar memory requirements in both systems, and so any deviation from this will indicate a difference between the worst case and expected times.

The experiments have been performed over a range of values for d, b, and the vector length. In order to further test our time complexity analysis we also varied the token vector and rule vector lengths independently, on the expectation that the comparative memory requirement would not vary.

The graphs in Fig. 2 are 3D scatter plots showing the memory requirement of the reduced ARCA as a percentage of the memory required of the original ARCA. The results clearly demonstrate that varying the relative token and rule lengths has very little effect on the comparative memory requirement.

It is also clear that the memory required by the reduced ARCA tends towards around 80% of that required by the original ARCA, which implies that the time complexity in the expected case is likely to be similarly improved.

Where the colour of a point is black, then the recall success rate for both architectures was at least 99%. As the colour of a point tends towards white (for the shortest vector lengths), it indicates that the reduced ARCA continued to achieve at least 99% recall success, but the original ARCA had a lower success rate. In no cases was the recall success rate of the original ARCA higher than that of the reduced ARCA, and so it can be concluded that the reduced ARCA improves the storage capacity of the network, even while reducing the memory

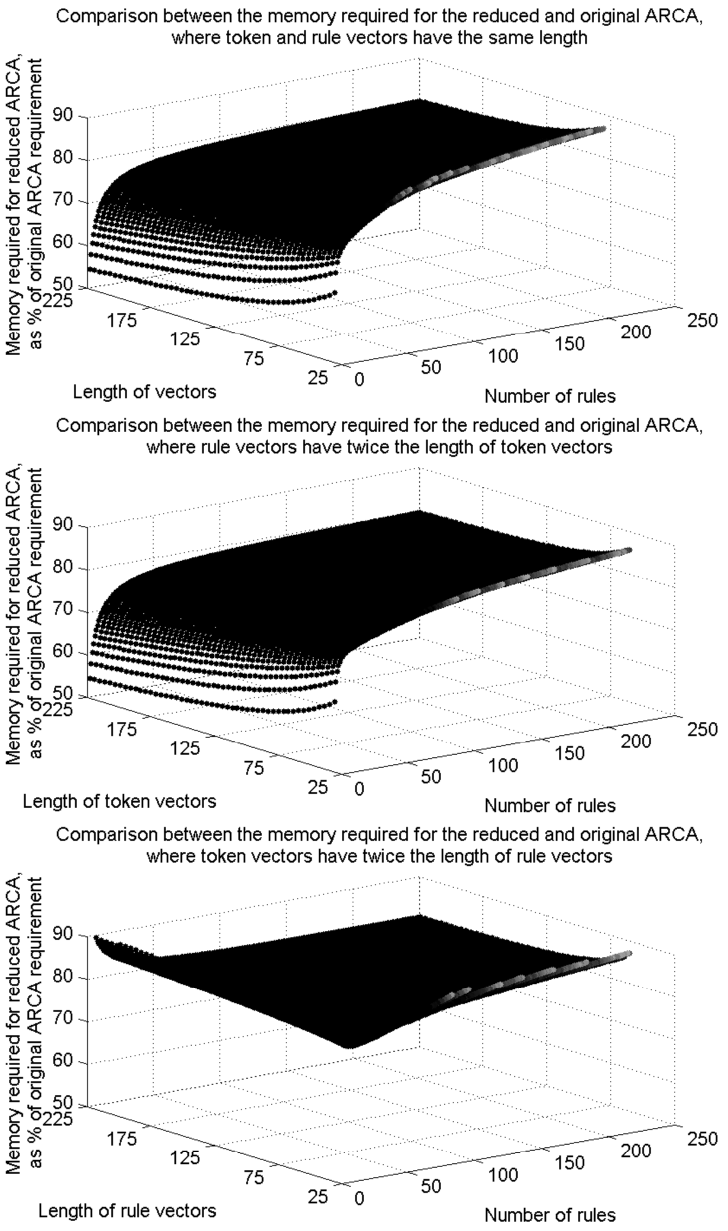


Fig. 2. 3D scatter plots showing the memory requirement of the reduced ARCA as a percentage of the memory requirement of the original ARCA. In the top plot, the length of token and rule vectors are equal. In the middle plot, the length of token vectors is double that of rule vectors. In the bottom plot, the length of rule vectors is double that of token vectors. For all points, the recall success rate for the reduced ARCA was at least 99%. As the colour tends from black to white, the recall success rate for the original ARCA moves from $\geq$99% towards 0%.

requirement. We chose to show the points at which recall success was at least 99%, as the nature of neural networks is that they may introduce a small amount of uncertainty. This cut-off, however, could be selected at any point—including 100%—with the trade-off that higher accuracy results in lower network capacity.

3 Conclusions and Further Work

This paper has described further improvements to the ARCA, with a simplified architecture which will aid in understanding, as well as reducing the memory requirements, the execution time, and improving the capacity.

Our analysis indicated that the performance of the original and the reduced architectures should be similar, if the token and rule vector weights are chosen to be equal. Our experimentation, on the other hand, clearly demonstrated that the reduced ARCA offers a 20% improvement over the original. This is due to the limitations of the worst case complexity, as previously explained. In order to improve the analysis, further work is required to create a better model of the interactions between vector pairs stored in a CMM. With such a model, the expected case could be analysed, based additionally on the number of rules stored in the system. This analysis will improve the theoretical basis for the application of ARCA to real rule-chaining problems.

References

1. Austin, J., Hobson, S., Burles, N., O'Keefe, S.: A Rule Chaining Architecture Using a Correlation Matrix Memory. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 49–56. Springer, Heidelberg (2012)
2. Kohonen, T.: Correlation Matrix Memories. IEEE Transactions on Computers, 353–359 (1972)
3. Gorodnichy, D.O.: Associative Neural Networks as Means for Low-Resolution Video-Based Recognition. IJCNN 2005, 3093–3098 (2005)
4. Ju, Q., O'Keefe, S., Austin, J.: Binary Neural Network Based 3D Facial Feature Localization. IJCNN 2009, 1462–1469 (2009)
5. Willshaw, D.J., Buneman, O.P., Longuet-Higgins, H.C.: Non-holographic Associative Memory. Nature 222, 960–962 (1969)
6. Ritter, H., Martinetz, T., Schulten, K., Barsky, D., Tesch, M., Kates, R.: Neural Computation and Self-Organizing Maps: An Introduction. Addison Wesley, Redwood City (1992)
7. Austin, J.: Parallel Distributed Computation in Vision. IEE Colloquium on Neural Networks for Image Processing Applications, 3/1–3/3 (1992)
8. Baum, E.B., Moody, J., Wilczek, F.: Internal Representations for Associative Memory. Biol. Cybernetics 59, 217–228 (1988)
9. Palm, G.: On the Storage Capacity of Associative Memories. In: Neural Assemblies, an Alternative Approach to Artificial Intelligence, pp. 192–199. Springer, New York (1982)

A Two-Stage Pretraining Algorithm for Deep Boltzmann Machines

KyungHyun Cho, Tapani Raiko, Alexander Ilin, and Juha Karhunen

Department of Information and Computer Science
Aalto University School of Science, Finland
{firstname.lastname}@aalto.fi

Abstract. A deep Boltzmann machine (DBM) is a recently introduced Markov random field model that has multiple layers of hidden units. It has been shown empirically that it is difficult to train a DBM with approximate maximum-likelihood learning using the stochastic gradient unlike its simpler special case, restricted Boltzmann machine (RBM). In this paper, we propose a novel pretraining algorithm that consists of two stages; obtaining approximate posterior distributions over hidden units from a simpler model and maximizing the variational lower-bound given the fixed hidden posterior distributions. We show empirically that the proposed method overcomes the difficulty in training DBMs from randomly initialized parameters and results in a better, or comparable, generative model when compared to the conventional pretraining algorithm.

Keywords: Deep Boltzmann Machine, Deep Learning, Pretraining.

1 Introduction

Deep Boltzmann machine (DBM), proposed in [14], is a recently introduced variant of Boltzmann machines which extends widely used restricted Boltzmann machines (RBM) to a model that has multiple hidden layers. It differs from the popular deep belief network (DBN) [5] in that every edge in the DBM model is undirected. In this way, DBMs facilitate propagating uncertainties across multiple layers of hidden variables.

Although it is straightforward to derive a learning algorithm for DBMs using a variational approximation and stochastic maximum likelihood method, recent research (see, for example, [14,4]) has shown that learning the parameters of DBMs is not trivial. Especially the generative performance of the trained model, commonly measured by the variational lower-bound of log-probabilities of test samples, tends to degrade as more hidden layers are added.

In [14] a greedy layer-wise pretraining algorithm was proposed to be used to initialize parameters of DBMs, and it was shown that it largely overcomes the difficulty of learning a good generative model.

Along this line of research, we propose another way to approach pretraining DBMs in this paper. The proposed scheme is based on an observation that training DBMs consists of two separate stages; approximating a posterior distribution over hidden units and updating parameters to maximize the lower-bound of log-likelihood given those states.

© Springer-Verlag Berlin Heidelberg 2013

Based on this observation, our proposed method pretrains a DBM in two stages. During the first stage we train a simpler, directed deep model such as DBNs or stacked denoising autoencoders (sDAE) to obtain an approximate posterior distribution over hidden units. With this fixed approximate posterior distribution, we train an RBM that learns a distribution over a combination of data samples and their corresponding posterior distributions of hidden units. Finetuning the model is then trivial as one only needs to free hidden variables from the approximate posterior distribution computed during the first stage.

We show that the proposed algorithm helps learning a good generative model which is empirically comparable to the pretraining method proposed in [14]. Furthermore, we discuss the potential degrees of freedom in extending the proposed approach.

2 Deep Boltzmann Machines

We start by describing deep Boltzmann machines (DBM) [14]. A DBM with L layers of hidden neurons is defined by the following energy function:

$$-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = \sum_{i}^{N_v} v_i b_i + \sum_{i}^{N_v} \sum_{j}^{N_1} v_i h_j^{(1)} w_{i,j} + \sum_{j}^{N_1} h_j^{(1)} c_j^{(1)} + \\ \sum_{l=2}^{L} \left(\sum_{j}^{N_l} h_j^{(l)} c_j^{(l)} + \sum_{j}^{N_l} \sum_{k}^{N_{l+1}} h_j^{(l)} h_k^{(l+1)} u_{j,k}^{(l)} \right), \tag{1}$$

where $\mathbf{v} = [v_i]_{i=1 \dots N_v}$ and $\mathbf{h}^{(l)} = \left[h_j^{(l)}\right]_{j=1 \dots N_l}$ are N_v binary visible units and N_l binary hidden units in the l-th hidden layer. $\mathbf{W} = [w_{i,j}]$ is the set of weights between the visible neurons and the first layer hidden neurons, while $\mathbf{U}^{(l)} = \left[u_{j,k}^{(l)}\right]$ is the set of weights between the l-th and $l+1$-th hidden neurons. b_i and $c_j^{(l)}$ are a bias to the i-th visible neuron and the j-th hidden neuron in the l-th hidden layer, respectively. We use $\boldsymbol{\theta}$ to denote a set of all these parameters.

With the energy function, a DBM can assign a probability to each state vector $\mathbf{x} = [\mathbf{v}; \mathbf{h}^{(1)}; \cdots; \mathbf{h}^{(L)}]$ using a Boltzmann distribution $p(\mathbf{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{-E(\mathbf{x} \mid \boldsymbol{\theta})\right\}$. Based on this property the parameters can be learned by maximizing the log-likelihood $\mathcal{L} = \sum_{n=1}^{N} \log \sum_{\mathbf{h}} p(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})$ given N training samples $\{\mathbf{v}^{(n)}\}_{n=1,\dots,N}$, where $\mathbf{h} = \left[\mathbf{h}^{(1)}; \cdots; \mathbf{h}^{(L)}\right]$.

The gradient computed by taking the partial derivative of the log-likelihood function with respect to each parameter is used in most cases with a mini-batch per update. It is then used to update the parameters, effectively forming a stochastic gradient ascent method. A standard way of computing gradient results in the following update rule for each parameter θ:

$$\nabla\theta = \left\langle -\frac{\partial E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})}{\partial \theta} \right\rangle_{\mathrm{d}} - \left\langle -\frac{\partial E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})}{\partial \theta} \right\rangle_{\mathrm{m}}, \tag{2}$$

where $\langle \cdot \rangle_{\mathrm{d}}$ and $\langle \cdot \rangle_{\mathrm{m}}$ denote the expectation over the data distribution $P(\mathbf{h} \mid \{\mathbf{v}^{(n)}\}, \boldsymbol{\theta})$ and the model distribution $P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})$, respectively [3].

3 Training Deep Boltzmann Machines

Although the update rules in Eq. (2) are well defined, it is intractable to compute them exactly. Hence, an approach that uses variational approximation together with Markov chain Monte Carlo (MCMC) sampling was proposed in [14].

First, the variational approximation is used to compute the expectation over the data distribution. It starts by approximating $p(\mathbf{h} \mid \mathbf{v}, \boldsymbol{\theta})$, which is intractable unless $L = 1$, by a factorial distribution $Q(\mathbf{h}) = \prod_{l=1}^{L} \prod_{j=1}^{N_l} \mu_j^{(l)}$. The variational parameters $\mu_j^{(l)}$ can then be estimated by the following fixed-point equation:

$$\mu_j^{(l)} \leftarrow f\left(\sum_{i=1}^{N_{l-1}} \mu_i^{(l-1)} w_{ij}^{(l-1)} + \sum_{k=1}^{N_{l+1}} \mu_k^{(l+1)} w_{kj}^{(l)} + c_j^{(l)} \right), \tag{3}$$

where $f(x) = \frac{1}{1+\exp\{-x\}}$. Note that $\mu_i^{(0)} = v_i$ and the update rule for the top layer does not contain the second summation term, that is $N_{L+1} = 0$.

This variational approximation provides the values of variational parameters that maximize the following lower-bound with respect to the current parameters:

$$p(\mathbf{v} \mid \boldsymbol{\theta}) \geq \mathbb{E}_{Q(\mathbf{h})}\left[-E(\mathbf{v}, \mathbf{h})\right] + \mathcal{H}(Q) - \log Z(\boldsymbol{\theta}), \tag{4}$$

where

$$\mathcal{H}(Q) = -\sum_{l=1}^{L} \sum_{j=1}^{N_l} \left(\mu_j^{(l)} \log \mu_j^{(l)} + (1 - \mu_j^{(l)}) \log(1 - \mu_j^{(l)}) \right)$$

is an entropy functional. Hence, each gradient update step does not increase the exact log-likelihood but its variational lower-bound.

Second, the expectation over the model distribution is computed by persistent sampling. The simplest approach is to use Gibbs sampling.

This approach closely resembles variational expectation-maximization (EM) algorithm (see, for example, [2]). Learning proceeds by alternating between finding the variational parameters $\boldsymbol{\mu}$ and updating the DBM parameters to maximize the given variational lower-bound using the stochastic gradient method. However, it has been known and will be shown in the experiments in this paper that training a DBM using this approach starting from randomly initialized parameters is not trivial [14,4].

Hence, in [14] a pretraining algorithm to initialize the parameters of DBMs was proposed. The pretraining algorithm greedily trains each layer of a DBM by considering each layer as an RBM, following a pretraining approach used for training deep belief networks (DBN) [5]. However, due to the undirectedness of edges in DBMs it has been proposed to use the first layer RBM with two duplicate copies of visible units with tied weights and the last layer RBM with two duplicate copies of hidden units with tied weights. Once one layer has been trained, another layer can be trained on the aggregate posterior distribution of the hidden units of the lower layer to extend the depth. After the pretraining, learned weights are used as initializations of weights of DBMs.

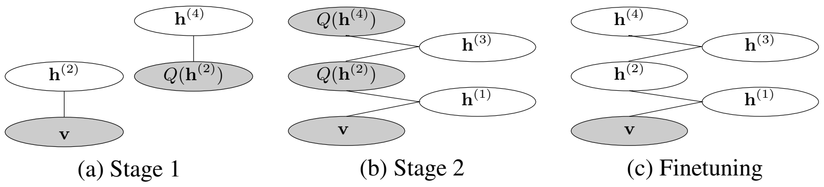


Fig. 1. Illustration of the two-stage pretraining algorithm followed by finetuning of all parameters. Shaded nodes indicate clamped variables whereas white nodes are free variables.

4 A Two-Stage Pretraining Algorithm

In this paper, we propose an alternative way of initializing parameters of a DBM compared with the one described at the end of Section 3. We employ an approach that separately obtains posterior distributions over hidden units and initializes parameters.

Before proceeding to the description of the proposed algorithm, we first divide the hidden layers of a DBM into two sets. Let us denote a vector of hidden units in the odd-numbered layers as $\mathbf{h}_+$ and the respective vector in the even-numbered layers as $\mathbf{h}_-$. In this sense we may define $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ as variational parameters of the hidden units in the odd-numbered layers and the even-number layers, respectively.

Stage 1: We focus on finding a good set of variational parameters $\boldsymbol{\mu}_-$ of $Q(\mathbf{h}_-)$ that has a potential to give a reasonably high variational lower-bound in Eq. (4). In other words, we propose to first find a good posterior distribution over hidden units given a visible vector regardless of parameter values of a DBM. Although it might sound unreasonable to find a good set of variational parameters without any fixed parameter values, we can do this by *borrowing* posterior distributions over latent variables from other models.

DBNs and sDAE's, described in [5] and [16], are natural choices to find a good approximate posterior distribution over units in the even-numbered hidden layers. One justification for using either of them is that they can be trained efficiently and well (see, e.g., [1] and references therein). It becomes a trivial task as one can iteratively train each even-numbered layer as either an RBM or a DAE on top of each other, as is a common practice when a DBN or a sDAE is trained.

Stage 2: Once a set of initial variational parameters $\boldsymbol{\mu}_-$ is found from a DBN or an sDAE, we train a model that has predictive power of the variational parameters given a visible vector. It can be simply done by letting an RBM learn a joint distribution of $\mathbf{v}$ and $\boldsymbol{\mu}_-$.

The structure of the RBM can be directly derived from the DBM such that its visible layer corresponds to the visible layer and the even-numbered hidden layers of the DBM and its hidden layer to the odd-numbered hidden layers of the DBM. The connections between them can also follow those of the DBM. This corresponds to finding a set of DBM parameters that *fit* the variational parameters obtained in the first stage.

Once an RBM has been trained, we can use the learned parameters as initializations for training the DBM, which corresponds to freeing $\mathbf{h}_-$ from its variational posterior distribution obtained in the first stage.

A simple illustration of the two-stage pretraining algorithm is given in Fig. 1.

4.1 Discussion

It is quite easy to see that the proposed algorithm has high degree of freedom to plug in alternative algorithms and models in both stages.

The most noticeable flexibility can be found in Stage 1. Any other machine learning model that gives reasonable posterior distributions over multiple layers of binary hidden units can be used instead of RBMs or DAEs. Also, instead of stacking each layer at a time, one could opt to train deep autoencoders at once using advanced backpropagation algorithms (see, for instance, [10]).

In Stage 2, one may opt to use a DAE instead of an RBM. It will make learning faster and therefore leave more time for finetuning the model afterward. Also, the use of different algorithms for training an RBM can be considered. For quicker pretraining, one may use contrastive divergence [6] , or for better initial models, advanced MCMC sampling methods could be used.

Another obvious possibility is to utilize the conventional pretraining algorithm proposed in [14] during the first stage. This approach gives approximate posterior distributions over all hidden units as well as initial values of the parameters. In this way, one may use either an RBM or a fully visible BM (FVBM) during the second stage starting from the initialized parameters. When an RBM is used in the second stage, one could simply discard $\boldsymbol{\mu}_+$.

One important point of the proposed algorithm is that it provides another research perspective in training DBMs. The existing pretraining scheme developed in [14,11] was based on the observation that under certain assumptions the variational lower-bound could be increased by learning weight parameters layer wise. However, the success of the proposed scheme suggests that it may not be the set of parameters that need to be directly pretrained, but the set of variational parameters that determine how tight the variational lower-bound is and their corresponding parameters.

5 Experiments

In the experiments, we train DBMs on two datasets which are a handwritten digit dataset (MNIST) [7] and Caltech-101 Silhouettes dataset [8]. We used the MNIST and Caltech-101 Silhouettes datasets because experimental results of using DBMs for both datasets are readily available for direct comparison [13,12,9].

We train DBMs with varying numbers of units in the hidden layers; 500-1000, 500-500-1000, 500-500-500-1000. The first two architectures were used in [13,12], which enables us to directly compare our proposed algorithm with the conventional pretraining algorithm.

For learning algorithms, we extensively tried various combinations. They are presented in Table 1. In summary, a DBM^{stage1}_{stage2} denotes a deep Boltzmann machine in which its superscript and subscript denote the algorithms used during the first and second stages, respectively.

We used contrastive divergence (CD) to train RBMs in the first stage, and the persistent CD [15] with coupled adaptive simulated annealing (CAST) was used in the second stage. DAEs were trained using stochastic backpropagation algorithm.

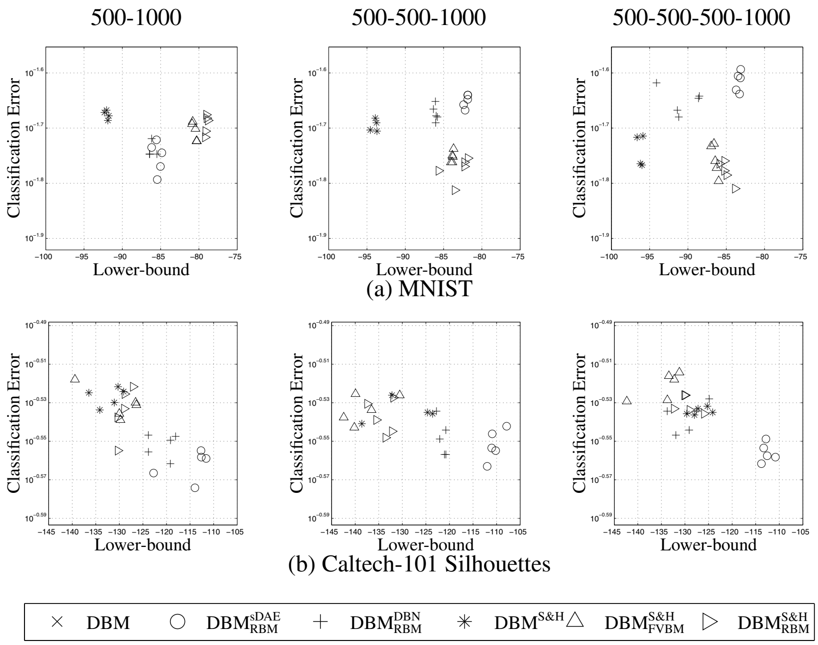


Fig. 2. Performance of the trained DBMs. Best performing models are in bottom right corners of each plot.

When a DBM was finetuned, we estimated the variational parameters by running at most 30 mean-field fixed-point updates. The model statistics, the negative part of the gradient, was computed by CAST.

Table 1. Algorithms used in the experiment. (S) – the pretraining algorithm from [14]

	Stage 1	Stage 2	Finetuning
DBM	×	×	DBM
$\text{DBM}^{\text{sDAE}}_{\text{RBM}}$	sDAE	RBM	DBM
$\text{DBM}^{\text{DBN}}_{\text{RBM}}$	DBN	RBM	DBM
$\text{DBM}^{\text{S\&H}}$	(S)	×	DBM
$\text{DBM}^{\text{S\&H}}_{\text{RBM}}$	(S)	RBM	DBM
$\text{DBM}^{\text{S\&H}}_{\text{FVBM}}$	(S)	FVBM	DBM

We evaluated the resulting models with the variational lower-bound of log-probabilities and the classification error of test samples. The variational lower-bounds reflect the generative performance of the model. The classification accuracy computed from a linear support vector machine (SVM) tells us the discriminative property of the hidden units. We trained a linear SVM for each hidden layer l using $\boldsymbol{\mu}_l$ as its features. This is expected to show how much information about input samples is captured by each hidden layer of the model.

All models were trained five times starting from different random initializations. We report medians over these random trials.

5.1 Result and Analysis

Fig. 2 presents the result using both the lower-bound of log-probabilities and the classification error of the test samples. As has already been expected, none of the models

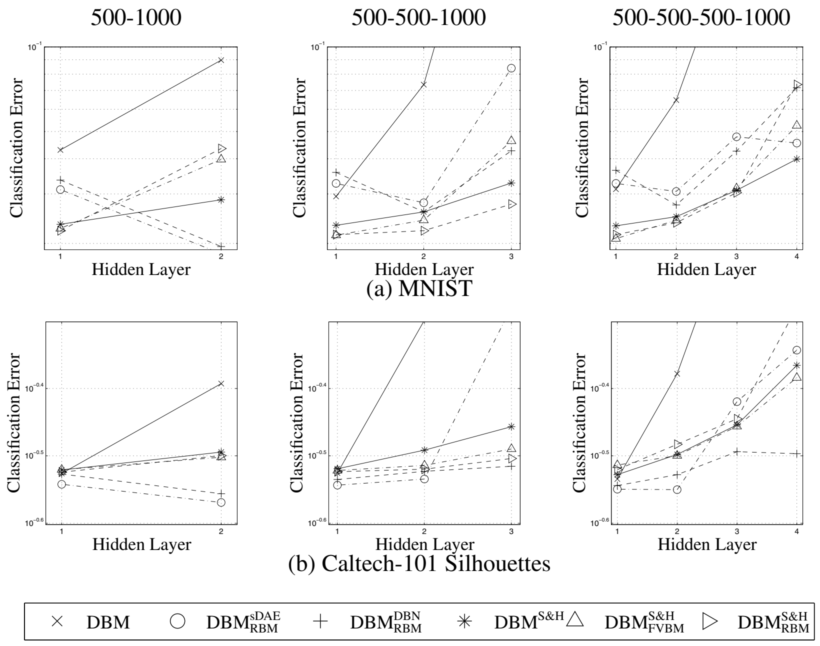


Fig. 3. Layer-wise Discriminative Performance. Lower is better.

trained *without* pretraining have been able to perform well enough to be presented inside the boundaries of the boxes in Fig. 2.

It is clear from the figures that the proposed two-stage pretraining algorithm outperforms, in all cases, the conventional pretraining algorithm (DBM$^{S\&H}$). On MNIST, the DBMs pretrained with the proposed algorithm using the conventional pretraining algorithm in the first stage achieved the best performance. In the case of Caltech-101 Silhouettes, DBM$^{sDAE}_{RBM}$ was able to achieve superior performance in both generative and discriminative modeling. It is notable that without any pretraining (DBM) we were not able to achieve any reasonable performance.

Fig. 3 presents layer-wise classification errors. It is clear from the significantly lower accuracies in the higher hidden layers of the DBMs trained without pretraining that pretraining is essential to allow upper layers to capture structures of data. DBM$^{DBN}_{RBM}$ and DBM$^{S\&H}_{RBM}$ were most effective in ensuring the upper hidden layers to have better discriminative property.

6 Conclusions

The experimental success of the proposed two-stage pretraining algorithm in training DBMs suggests that the difficulty of DBM learning might be due to the fact that the estimated variational lower-bound at the initial stage of learning is too crude, or too loose. Once one initializes the variational parameters well enough by utilizing another deep hierarchical model, the parameters of a DBM can be fitted to give a tighter variational lower-bound which facilitates jointly estimating all parameters.

The proposed two-stage pretraining algorithm provides a general framework in which many hierarchical deep learning models can be used. It even makes possible to include the conventional pretraining algorithm as a part of the proposed algorithm and improve upon it. This is a significant step in developing and improving a training algorithm for DBMs, as it allows us to fully utilize other learning algorithms that have been extensively studied previously.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation Learning: A Review and New Perspectives. arXiv:1206.5538 [cs.LG] (June 2012)
2. Bishop, C.M.: Pattern Recognition and Machine Learning, corrected 2nd printing edn. Springer (2007)
3. Cho, K.: Improved Learning Algorithms for Restricted Boltzmann Machines. Master's thesis, Aalto University School of Science (2011)
4. Desjardins, G., Courville, A., Bengio, Y.: On training deep Boltzmann machines. arXiv:1203.4416 [cs.NE] (March 2012)
5. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
6. Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771–1800 (2002)
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (11), 2278–2324
8. Marlin, B.M., Swersky, K., Chen, B., de Freitas, N.: Inductive principles for restricted Boltzmann machine learning. In: Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2010), pp. 509–516 (2010)
9. Montavon, G., Müller, K.-R.: Deep Boltzmann machines and the centering trick. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) NN: Tricks of the Trade, 2nd edn. LNCS, vol. 7700, pp. 621–637. Springer, Heidelberg (2012)
10. Raiko, T., Valpola, H., LeCun, Y.: Deep learning made easier by linear transformations in perceptrons. In: Proc. of the 15th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2012), La Palma, Canary Islands, Spain (April 2012)
11. Salakhutdinov, R., Hinton, G.E.: A Better Way to Pre-Train Deep Boltzmann Machines. In: Advances in Neural Information Processing Systems (2012)
12. Salakhutdinov, R.: Learning deep Boltzmann machines using adaptive MCMC. In: Fürnkranz, J., Joachims, T. (eds.) Proc. of the 27th Int. Conf. on Machine Learning (ICML 2010), pp. 943–950. Omnipress, Haifa (2010)
13. Salakhutdinov, R., Hinton, G.: An efficient learning procedure for deep Boltzmann machines. Tech. Rep. MIT-CSAIL-TR-2010-037, MIT (August 2010)
14. Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: Proc. of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2009), pp. 448–455 (2009)
15. Tieleman, T., Hinton, G.E.: Using fast weights to improve persistent contrastive divergence. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 1033–1040. ACM, New York (2009)
16. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11, 3371–3408 (2010)

A Low-Energy Implementation of Finite Automata by Optimal-Size Neural Nets

Jiří Šíma[⋆]

Institute of Computer Science, Academy of Sciences of the Czech Republic,
P. O. Box 5, 18207 Prague 8, Czech Republic
sima@cs.cas.cz

Abstract. Recently, a new so-called energy complexity measure has been introduced and studied for feedforward perceptron networks. This measure is inspired by the fact that biological neurons require more energy to transmit a spike than not to fire and the activity of neurons in the brain is quite sparse, with only about 1% of neurons firing. We investigate the energy complexity for recurrent networks which bounds the number of active neurons at any time instant of a computation. We prove that any deterministic finite automaton with m states can be simulated by a neural network of optimal size $s = \Theta(\sqrt{m})$ with time overhead $O(s/e)$ per one input bit, using the energy $O(e)$, for any $e = \Omega(\log s)$ and $e = O(s)$, which shows the time-energy tradeoff in recurrent networks.

1 Introduction

In biological neural networks the energy cost of a firing neuron is relatively high while energy supplied to the brain is limited and hence the activity of neurons in the brain is quite sparse, with only about 1% of neurons firing [4]. This is in contrast to artificial neural networks in which on average every second unit fires during a computation. This fact has recently motivated the definition of a new complexity measure for *feedforward* perceptron networks (threshold circuits), the so-called energy complexity [11] which is the maximum number of units in the network which output 1, taken over all the inputs to the circuit. The energy has been shown to be closely related by tradeoff results to other complexity measures such as the network size (i.e., the number of neurons) [13, 15], the circuit depth (i.e., parallel computational time) [12, 13], and the fan-in (i.e., the maximum number of inputs to a single unit) [10] etc. In addition, energy complexity has found its use in circuit complexity, e.g. as a tool for proving the lower bounds [14] etc.

In this paper, we investigate for the first time energy complexity for *recurrent* neural networks which we define to be the maximum number of neurons outputting 1 at any time instant, taken over all possible computations. It has been known for a long time that the computational power of binary-state recurrent networks corresponds to that of finite automata since the network of size s units can reach only a finite number (at most 2^s) different states [8]. A simple way of simulating a given deterministic finite automaton A with m states by a neural

⋆ Research was supported by the projects GA ČR P202/10/1333 and RVO: 67985807.

© Springer-Verlag Berlin Heidelberg 2013

network N of size $O(m)$ is to implement each of the $2m$ transitions of A (having 0 and 1 transitions for each state) by a single unit in N which checks whether the input bit agrees the respective type of transition [6]. Clearly, this simple linear-size implementation of finite automata requires only a constant energy.

Much effort was given to reducing the size of neural automata (e.g. [1–3, 9]), and indeed, neural networks of size $\Theta(\sqrt{m})$ implementing a given deterministic finite automaton with m states were proposed and proven to be size-optimal [2, 3]. A natural question arises: what is the energy consumption when simulating finite automata by optimal-size neural networks? We answer this question by proving the tradeoff between the energy and the time overhead of the simulation. In particular, we prove that an optimal-size neural network of $s = \Theta(\sqrt{m})$ units can be constructed to simulate a deterministic finite automaton with m states using the energy $O(e)$ for any $e = \Omega(\log s)$ and $e = O(s)$, while the time overhead for processing one input bit is $O(s/e)$. For this purpose, we adapt the asymptotically optimal method of threshold circuit synthesis due to Lupanov [5].

This paper is organized as follows. In Section 2, the main result is formulated after a brief review of the basic definitions. The subsequent two sections are devoted to the technical proof: Section 3 deals with a decomposition of the transition function and Section 4 describes the construction of low-energy neural automata. Section 5 concludes with some remarks on lower bounds on the energy complexity of neural network automata.

2 Neural Networks as Finite Automata

We will first specify the model of a recurrent *neural network* N. The network consists of s *units (neurons)*, indexed as $V = \{1, \ldots, s\}$, where s is called the network *size*. The units are connected into an oriented graph representing the *architecture* of N, in which each edge (i, j) leading from unit i to j is labeled with an integer *weight* $w(i, j)$. The absence of a connection within the architecture corresponds to a zero weight between the respective neurons, and vice versa.

The *computational dynamics* of N determines for each unit $j \in V$ its binary *state (output)* $y_j^{(t)} \in \{0, 1\}$ at discrete time instants $t = 0, 1, 2, \ldots$. We say that neuron j is *active (fires)* at time t if $y_j^{(t)} = 1$, while j is *passive* for $y_j^{(t)} = 0$. This establishes the *network state* $\mathbf{y}^{(t)} = (y_1^{(t)}, \ldots, y_s^{(t)}) \in \{0, 1\}^s$ at each discrete time instant $t \geq 0$. At the beginning of a computation, N is placed in an *initial state* $\mathbf{y}^{(0)}$. At discrete time instant $t \geq 0$, an excitation of any neuron $j \in V$ is defined as $\xi_j^{(t)} = \sum_{i=1}^{s} w(i, j) y_i^{(t)} - h(j)$ including an integer *threshold* $h(j)$ local to unit j. At the next instant $t + 1$, the neurons $j \in \alpha_{t+1}$ from a selected subset $\alpha_{t+1} \subseteq V$ update their states $y_j^{(t+1)} = H(\xi_j^{(t)})$ in parallel by applying the *Heaviside function* $H(\xi)$ which is defined to be 1 for $\xi \geq 0$ and 0 for $\xi < 0$. The remaining units $j \in V \setminus \alpha_{t+1}$ do not change their outputs, that is $y_j^{(t+1)} = y_j^{(t)}$ for $j \notin \alpha_{t+1}$. In this way, the new network state $\mathbf{y}^{(t+1)}$ at time $t + 1$ is determined. We define the *energy complexity* of N to be the maximum number of active units $\sum_{j=1}^{s} y_j^{(t)}$ at any time instant $t \geq 0$, taken over all computations of N.

The computational power of recurrent neural networks has been studied analogously to the traditional models of computations so that the networks are exploited as acceptors of formal languages $L \subseteq \{0,1\}^*$ over the binary alphabet. For the finite networks that are to recognize regular languages, the following input/output protocol has been used [1–3, 7–9]. A binary input word (string) $\mathbf{x} = x_1 \dots x_n \in \{0,1\}^n$ of arbitrary length $n \geq 0$ is sequentially presented to the network bit by bit via an *input neuron* in $\in V$. The state of this unit is externally set (and clamped) to the respective input bits at prescribed time instants, regardless of any influence from the remaining neurons in the network, that is, $y_{\text{in}}^{(\tau(i-1))} = x_i$ for $i = 1, \dots, n$ where an integer parameter $\tau \geq 1$ is the *period* or *time overhead* for processing a single input bit. Then, an *output neuron* out $\in V$ signals at time τn whether the input word belongs to underlying language L, that is, $y_{\text{out}}^{(\tau n)} = 1$ for $\mathbf{x} \in L$, whereas $y_{\text{out}}^{(\tau n)} = 0$ for $\mathbf{x} \notin L$.

Now, we can formulate our main result concerning a low-energy implementation of finite automata by optimal-size neural nets:

Theorem 1. *A given deterministic finite automaton A with m states can be simulated by a neural network N of optimal size $s = \Theta(\sqrt{m})$ neurons with time overhead $O(s/e)$ per one input bit, using the energy $O(e)$, where e is any function satisfying $e = \Omega(\log s)$ and $e = O(s)$.*

Proof. A set Q of m states of a given deterministic finite automaton A can be arbitrarily enumerated so that each $q \in Q$ is binary encoded using $p = \lceil \log m \rceil + 1$ bits including one additional bit which indicates the final states. Then, the respective transition function $\delta : Q \times \{0,1\} \longrightarrow Q$ of A, producing its new state $q_{\text{new}} = \delta(q_{\text{old}}, x) \in Q$ from the old state $q_{\text{old}} \in Q$ and input bit $x \in \{0,1\}$, can be viewed as a vector Boolean function $\mathbf{f} : \{0,1\}^{p+1} \longrightarrow \{0,1\}^p$ in terms of binary encoding of states. In the following two sections we will adapt the asymptotically optimal method of threshold circuit synthesis due to Lupanov [5] to implement $\mathbf{f}$ by a low-energy recurrent neural network.

3 The Transition Function Decomposition

The $p+1$ arguments of vector function $\mathbf{f}(\mathbf{u}, \mathbf{v}, \mathbf{z})$ are split into three groups $\mathbf{u} = (u_1, \dots, u_{p_1})$, $\mathbf{v} = (v_1, \dots, v_{p_2})$, and $\mathbf{z} = (z_1, \dots, z_{p_3})$, respectively, where $p_1 = \lfloor (p + 1 - \log p - \log(p + 1 - \log p))/2 \rfloor$, $p_3 = \lfloor \log(p + 1 - \log p) - 2 \rfloor$, and $p_2 = p + 1 - p_3 - p_1$. Then, each function element $f_k : \{0,1\}^{p+1} \longrightarrow \{0,1\}$ $(1 \leq k \leq p)$ of vector function $\mathbf{f} = (f_1, \dots, f_p)$ is decomposed to

$$f_k(\mathbf{u}, \mathbf{v}, \mathbf{z}) = \bigvee_{\mathbf{c} \in \{0,1\}^{p_3}} \left(f_k(\mathbf{u}, \mathbf{v}, \mathbf{c}) \wedge \bigwedge_{j=1}^{p_3} \ell_{c_j}(z_j) \right) , \tag{1}$$

where the respective literals are defined as $\ell_c(z) = z$ for $c = 1$ and $\ell_c(z) = \neg z$ for $c = 0$. Furthermore, we define vector functions $\mathbf{g}_k : \{0,1\}^{p_1 + p_2} \longrightarrow \{0,1\}^{p_1}$ for $k = 1, \dots, p$ as

$$\mathbf{g}_k(\mathbf{u}, \mathbf{v}) = (f_k(\mathbf{u}, \mathbf{v}, [0]^{p_3}), f_k(\mathbf{u}, \mathbf{v}, [1]^{p_3}), \dots, f_k(\mathbf{u}, \mathbf{v}, [2^{p_3} - 1]^{p_3}), 0, \dots, 0) \tag{2}$$

where $[j]^n = \mathbf{c} = (c_1, \ldots, c_n) \in \{0,1\}^n$ denotes an n-bit binary representation of integer $j \geq 0$, that is, $j = \langle \mathbf{c} \rangle = \sum_{i=1}^{n} 2^{i-1} c_i$. The vector produced by $\mathbf{g}_k$ in (2) has p_1 elements out of which the first 2^{p_3} items are defined using f_k for all possible values of argument $\mathbf{z} \in \{0,1\}^{p_3}$, while the remaining ones are 0s, which is a correct definition since $2^{p_3} < p_1$ for sufficiently large p.

Denote $r = p_1 - 1$. For each $\mathbf{g}_k$ ($1 \leq k \leq p$), we will construct four vector functions $\mathbf{g}_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0,1\}^{p_1}$ and $\mathbf{h}_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0,1\}^{p_1}$ for $a \in \{0,1\}$ such that

$$\mathbf{g}_k(a, \mathbf{u}', \mathbf{v}) = \mathbf{g}_k^a(\mathbf{u}', \mathbf{v}) \oplus \mathbf{h}_k^a(\mathbf{u}', \mathbf{v}) \tag{3}$$

for any $a \in \{0,1\}$, $\mathbf{u}' \in \{0,1\}^r$, and $\mathbf{v} \in \{0,1\}^{p_2}$, where $\oplus$ denotes a bitwise parity (i.e., $\mathbf{z} = \mathbf{x} \oplus \mathbf{y} \in \{0,1\}^n$ is defined for vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, $\mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$, and $\mathbf{z} = (z_1, \ldots, z_n) \in \{0,1\}^n$ as $z_i = 1$ iff $x_i \neq y_i$ for every $i = 1, \ldots, n$) which is an associative operation. In addition, the construction will guarantee that for any $a \in \{0,1\}$, $\mathbf{v} \in \{0,1\}^{p_2}$, and $\mathbf{u}'_1, \mathbf{u}'_2 \in \{0,1\}^r$,

$$\text{if } \mathbf{u}'_1 \neq \mathbf{u}'_2\,, \text{ then } \mathbf{g}_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \mathbf{g}_k^a(\mathbf{u}'_2, \mathbf{v}) \text{ and } \mathbf{h}_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \mathbf{h}_k^a(\mathbf{u}'_2, \mathbf{v})\,. \tag{4}$$

For any $\mathbf{v} \in \{0,1\}^{p_2}$, the function values of $\mathbf{g}_k^a$ are defined inductively as $\mathbf{g}_k^a([i]^r, \mathbf{v}) \in \{0,1\}^{p_1} \setminus G_k^a(i, \mathbf{v})$ is chosen arbitrarily for $i = 0, \ldots, 2^r - 1$ where

$$\begin{aligned} G_k^a(i, \mathbf{v}) = \{&\mathbf{g}_k^a([j]^r, \mathbf{v})\,, \\ &\mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k(a, [j]^r, \mathbf{v}) \oplus \mathbf{g}_k^a([j]^r, \mathbf{v}) \mid j = 0, \ldots, i-1\}\ , \end{aligned} \tag{5}$$

and functions $\mathbf{h}_k^a$ are defined so that equation (3) is met:

$$\mathbf{h}_k^a(\mathbf{u}', \mathbf{v}) = \mathbf{g}_k(a, \mathbf{u}', \mathbf{v}) \oplus \mathbf{g}_k^a(\mathbf{u}', \mathbf{v})\,. \tag{6}$$

Note that $\emptyset = G_k^a(0, \mathbf{v}) \subseteq G_k^a(1, \mathbf{v}) \subseteq \cdots \subseteq G_k^a(2^r - 1, \mathbf{v})$ and $|G_k^a(i, \mathbf{v})| \leq 2i$ according to (5), which implies $|G_k^a(i, \mathbf{v})| \leq |G_k^a(2^r - 1, \mathbf{v})| \leq 2(2^r - 1)$. Hence, $|\{0,1\}^{p_1} \setminus G_k^a(i, \mathbf{v})| \geq 2^{p_1} - 2(2^r - 1) = 2$, which ensures that $\mathbf{g}_k^a(\mathbf{u}', \mathbf{v})$ is correctly defined for all arguments $\mathbf{u}' \in \{0,1\}^r$. Moreover, condition (4) is satisfied because for any $i, j \in \{0, \ldots, 2^r - 1\}$ such that $i > j$, definition (5) secures $\mathbf{g}_k^a([i]^r, \mathbf{v}) \neq \mathbf{g}_k^a([j]^r, \mathbf{v})$ and $\mathbf{h}_k^a([i]^r, \mathbf{v}) = \mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k^a([i]^r, \mathbf{v}) \neq \mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k(a, [j]^r, \mathbf{v}) \oplus \mathbf{g}_k^a([j]^r, \mathbf{v}) = \mathbf{h}_k^a([j]^r, \mathbf{v})$ by using (6) and the fact that $\mathbf{x} \oplus \mathbf{x} \oplus \mathbf{y} = \mathbf{y}$.

We further decompose $\mathbf{g}_k^a$ and $\mathbf{h}_k^a$ by using the functions $\varphi_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0, \ldots, 2^{p_1} - 1\}$ and $\psi_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0, \ldots, 2^{p_1} - 1\}$ as

$$\mathbf{g}_k^a(\mathbf{u}', \mathbf{v}) = [\varphi_k^a(\mathbf{u}', \mathbf{v})]^{p_1} \ \text{ and } \ \mathbf{h}_k^a(\mathbf{u}', \mathbf{v}) = [\psi_k^a(\mathbf{u}', \mathbf{v})]^{p_1}\ , \tag{7}$$

respectively, which satisfy for any $a \in \{0,1\}$, $\mathbf{v} \in \{0,1\}^{p_2}$, and $\mathbf{u}'_1, \mathbf{u}'_2 \in \{0,1\}^r$,

$$\text{if } \mathbf{u}'_1 \neq \mathbf{u}'_2\,, \text{ then } \varphi_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \varphi_k^a(\mathbf{u}'_2, \mathbf{v}) \text{ and } \psi_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \psi_k^a(\mathbf{u}'_2, \mathbf{v}) \tag{8}$$

according to (4). Now, we can plug (2), (3), and (7) into (1) which results in

$$f_k(a, \mathbf{u}', \mathbf{v}, \mathbf{z}) = \bigvee_{\mathbf{c} \in \{0,1\}^{p_3}} \left((\mathbf{g}_k(a, \mathbf{u}', \mathbf{v}))_{\langle \mathbf{c} \rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i) \right)$$

$$= \bigvee_{\mathbf{c}\in\{0,1\}^{p_3}} \left(\left(\left([\varphi_k^a(\mathbf{u}',\mathbf{v})]^{p_1}\right)_{\langle\mathbf{c}\rangle} \wedge \neg \left([\psi_k^a(\mathbf{u}',\mathbf{v})]^{p_1}\right)_{\langle\mathbf{c}\rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i) \right) \right.$$
$$\left. \vee \left(\neg \left([\varphi_k^a(\mathbf{u}',\mathbf{v})]^{p_1}\right)_{\langle\mathbf{c}\rangle} \wedge \left([\psi_k^a(\mathbf{u}',\mathbf{v})]^{p_1}\right)_{\langle\mathbf{c}\rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i) \right) \right) , \qquad (9)$$

where $(\mathbf{x})_i$ denotes the ith element of vector $\mathbf{x}$.

4 The Finite Automaton Implementation

In this section, we will describe the construction of low-energy recurrent neural network N simulating a given finite automaton A. In particular, set of neurons V is composed of four disjoint layers $V = \nu_0 \cup \nu_1 \cup \nu_2 \cup \nu_3$. A current state of A and an input bit are stored using $p+1$ neurons which constitute layer ν_0. Thus, set ν_0 includes the input neuron in $\in \nu_0$ and the output neuron out $\in \nu_0$ which saves the bit (in the state encoding) that indicates the final states. We will implement formula (9) in N for evaluating the transition function $\mathbf{f}$ in terms of binary encoding of states in order to compute the new state of A. Layer $\nu_0 = \{\text{in}\} \cup \nu_{01} \cup \nu_{02} \cup \nu_{03}$ is disjointly split into four parts corresponding to the partition of arguments of $\mathbf{f}(a, \mathbf{u}', \mathbf{v}, \mathbf{z})$, respectively, that is, $\nu_{01} = \{u_1, \ldots, u_r\}$, $\nu_{02} = \{v_1, \ldots, v_{p_2}\}$, and $\nu_{03} = \{z_1, \ldots, z_{p_3}\}$.

The next layer $\nu_1 = \nu_{11} \cup \nu_{12}$ consists of 2^{p_2} neurons in $\nu_{11} = \{\mu_{\langle\mathbf{b}\rangle} \,|\, \mathbf{b} \in \{0,1\}^{p_2}\}$ for computing all possible monomials $\bigwedge_{i=1}^{p_2} \ell_{b_i}(v_i)$ over input variables $\mathbf{v}$, and two control units in $\nu_{12} = \{\kappa_0^0, \kappa_0^1\}$ which indicate the input bit value. This is implemented by weights $w(v_i, \mu_{\langle\mathbf{b}\rangle}) = 2b_i - 1$ for $i = 1, \ldots, p_2$, and threshold $h(\mu_{\langle\mathbf{b}\rangle}) = \sum_{i=1}^{p_2} b_i$, for any $\mathbf{b} = (b_1, \ldots, b_{p_2}) \in \{0,1\}^{p_2}$ so that $\mu_{\langle\mathbf{b}\rangle}$ fires iff $\mathbf{b} = \mathbf{v}$. In addition, we define $w(\text{in}, \kappa_0^1) = -w(\text{in}, \kappa_0^0) = 1$ and $h(\kappa_0^1) = 1$, $h(\kappa_0^0) = 0$, which ensures that $y_{in} = 1$ iff κ_0^1 fires iff κ_0^0 is passive.

Furthermore, layer $\nu_2 = \nu_{21} \cup \nu_{22}$ where $\nu_{21} = \{\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a}, \gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a} \,|\, 1 \le k \le p\,, a \in \{0,1\}\,, j = 0, \ldots, 2^{p_1} - 1\}$, and $\nu_{22} = \{\kappa_i^a \,|\, a \in \{0,1\}\,, i = 1, \ldots, d+1\}$ with $d = \lceil 2p2^{p_1}/e \rceil$, serves for a low-energy computation of functions $\varphi_k^a(\mathbf{u}', \mathbf{v})$ and $\psi_k^a(\mathbf{u}', \mathbf{v})$. We will first show how to implement functions $\varphi_k^a(\mathbf{u}', \mathbf{v})$ for any $1 \le k \le p$ and $a \in \{0,1\}$ with no constraints on energy by using the outputs of neurons from ν_{01} and ν_{11}. In particular, 2^{p_1} pairs of neurons $\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a} \in \nu_{21}$ for $j = 0, \ldots, 2^{p_1} - 1$ are employed having zero thresholds for now (their thresholds will be defined below for the low-energy implementation) and weights $w(u_i, \gamma_{kj}^{\varphi a}) = -w(u_i, \lambda_{kj}^{\varphi a}) = 2^{i-1}$ for $i = 1, \ldots, r$ and $w(\mu_{\langle\mathbf{b}\rangle}, \lambda_{kj}^{\varphi a}) = -w(\mu_{\langle\mathbf{b}\rangle}, \gamma_{kj}^{\varphi a}) = d_{kj}^{\varphi a\mathbf{b}} \in \{0, \ldots, 2^r - 1\}$ such that $j = \varphi_k^a([d_{kj}^{\varphi a\mathbf{b}}]^r, \mathbf{b}) \in \{0, \ldots, 2^{p_1} - 1\}$ for $\mathbf{b} \in \{0,1\}^{p_2}$. Note that $d_{kj}^{\varphi a\mathbf{b}}$ is uniquely defined according to (8). It follows that for given $\mathbf{u}' \in \{0,1\}^r$ and $\mathbf{v} \in \{0,1\}^{p_2}$, neuron $\gamma_{kj}^{\varphi a}$ fires iff $\sum_{i=1}^r w(u_i, \gamma_{kj}^{\varphi a}) y_{u_i} + \sum_{\mathbf{b}\in\{0,1\}^{p_2}} w(\mu_{\langle\mathbf{b}\rangle}, \gamma_{kj}^{\varphi a}) y_{\mu_{\langle\mathbf{b}\rangle}} \ge 0$ iff $\sum_{i=1}^r 2^{i-1} u_i' - d_{kj}^{\varphi a\mathbf{v}} \ge 0$ iff $\langle\mathbf{u}'\rangle \ge d_{kj}^{\varphi a\mathbf{v}}$, since $y_{\mu_{\langle\mathbf{b}\rangle}} = 1$ iff $\mathbf{b} = \mathbf{v}$. Similarly, neuron $\lambda_{kj}^{\varphi a}$ is active iff $\langle\mathbf{u}'\rangle \le d_{kj}^{\varphi a\mathbf{v}}$. Hence, both neurons $\gamma_{kj}^{\varphi a}$ and $\lambda_{kj}^{\varphi a}$ fire at the same time iff $\langle\mathbf{u}'\rangle = d_{kj}^{\varphi a\mathbf{v}}$ iff $j = \varphi_k^a(\mathbf{u}', \mathbf{v})$, which implements function $\varphi_k^a(\mathbf{u}', \mathbf{v})$. Functions $\psi_k^a(\mathbf{u}', \mathbf{v})$ for any $1 \le k \le p$ and

$a \in \{0,1\}$ are implemented analogously (replace φ by ψ above) using 2^{p_1} pairs of neurons $\gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a} \in \nu_{21}$ for $j = 0, \ldots, 2^{p_1}-1$, that is, both units $\gamma_{kj}^{\psi a}$ and $\lambda_{kj}^{\psi a}$ are active iff $j = \psi_k^a(\mathbf{u}', \mathbf{v})$.

We employ control units $\kappa_i^a \in \nu_{12} \cup \nu_{22}$ for $a \in \{0,1\}$ and $i = 0, \ldots, d+1$, for synchronizing the computation of functions $\varphi_k^a(\mathbf{u}', \mathbf{v})$, $\psi_k^a(\mathbf{u}', \mathbf{v})$ by neurons from ν_{21} so that their energy consumption is bounded by $e+2$. For this purpose, we split set $\nu_{21} = \nu_{21}^0 \cup \nu_{21}^1$ into two parts $\nu_{21}^a = \{\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a}, \gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a} \,|\, 1 \le k \le p\,,\, j = 0, \ldots, 2^{p_1}-1\}$ of size $4p2^{p_1}$ according to $a \in \{0,1\}$, and each such part is further partitioned into d blocks of size at most $2e$, that is $\nu_{21}^a = \bigcup_{i=1}^d \beta_i^a$ where $|\beta_i^a| \le 2e$. In addition, we require for every $i = 1, \ldots, d$, if $\gamma_{kj}^{\varphi a} \in \beta_i^a$, then $\lambda_{kj}^{\varphi a} \in \beta_i^a$, and if $\gamma_{kj}^{\psi a} \in \beta_i^a$, then $\lambda_{kj}^{\psi a} \in \beta_i^a$. For any $1 \le i \le d$ and $a \in \{0,1\}$, the neurons in block β_i^a are activated by control unit κ_{i-1}^a using the weights $w(\kappa_{i-1}^a, j) = W$ for all $j \in \beta_i^a$, while all neurons $j \in \nu_{21}$ are blocked by thresholds $h(j) = W$ where $W = 2^r$ if there is no support from a corresponding control unit. For current input bit $y_{\text{in}} = a \in \{0,1\}$, the control units $\kappa_0^a, \ldots, \kappa_{d+1}^a$ fire successively one by one, which is achieved by weights $w(\kappa_i^a, \kappa_{i+1}^a) = 1$ for $i = 0, \ldots, d$, $w(\kappa_i^a, \kappa_0^a) = -1$ for $i = 0, \ldots, d+1$, and thresholds $h(\kappa_i^a) = 1$ for $i = 1, \ldots, d+1$. This ensures that only the neurons from one block β_i^a of size at most $2e$ can fire at the same time. In fact, we know that just one unit of each pair $\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a} \in \beta_i^a$ or $\gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a} \in \beta_i^a$ is active except for the special pairs of both firing units $\gamma_{kj_\varphi}^{\varphi a}, \lambda_{kj_\varphi}^{\varphi a}$ and $\gamma_{kj_\psi}^{\psi a}, \lambda_{kj_\psi}^{\psi a}$ such that $\varphi_k^a(\mathbf{u}', \mathbf{v}) = j_\varphi$ and $\psi_k^a(\mathbf{u}', \mathbf{v}) = j_\psi$, respectively. Hence, the energy consumption of ν_{21} is bounded by $e+2$. Finally, we must also guarantee that the resulting function values $\varphi_k^a(\mathbf{u}', \mathbf{v}) = j_\varphi$, $\psi_k^a(\mathbf{u}', \mathbf{v}) = j_\psi$ are stored, that is, neurons $\gamma_{kj_\varphi}^{\varphi a}, \lambda_{kj_\varphi}^{\varphi a}, \gamma_{kj_\psi}^{\psi a}, \lambda_{kj_\psi}^{\psi a}$ remain active without any support from corresponding control units until all blocks perform computation which is indicated by control unit κ_{d+1}^a. Neuron κ_{d+1}^a then resets all neurons in ν_{21} before becoming itself passive. This is implemented by symmetric weights $w(\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a}) = w(\lambda_{kj}^{\varphi a}, \gamma_{kj}^{\varphi a}) = w(\gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a}) = w(\lambda_{kj}^{\psi a}, \gamma_{kj}^{\psi a}) = W$ for $a \in \{0,1\}$, $k = 1, \ldots, p$, $j = 0, \ldots, 2^{p_1}-1$, and $w(\kappa_{d+1}^a, j) = -W$ for all $j \in \nu_{21}$.

Finally, layer $\nu_3 = \{\pi_{k\langle \mathbf{c} \rangle}, \varrho_{k\langle \mathbf{c} \rangle} \,|\, 1 \le k \le p\,,\, \mathbf{c} \in \{0,1\}^{p_3}\}$ is composed of 2^{p_3} pairs of neurons $\pi_{k\langle \mathbf{c} \rangle}, \varrho_{k\langle \mathbf{c} \rangle}$ for each $k = 1, \ldots, p$ which compute $([\varphi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle}$ $\wedge \neg([\psi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i)$ and $\neg([\varphi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle} \wedge ([\psi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle} \wedge$ $\bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i)$ from (9), respectively, for current input $y_{\text{in}} = a \in \{0,1\}$ by using the states of neurons from ν_{03} and the outputs of units $\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a}, \gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a} \in \nu_{21}$ for $j = 0, \ldots, 2^{p_1}-1$ after κ_{d+1}^a fires. For $\mathbf{c} \in \{0,1\}^{p_3}$, we define weights $w(\gamma_{kj}^{\varphi a}, \pi_{k\langle \mathbf{c} \rangle}) = w(\lambda_{kj}^{\varphi a}, \pi_{k\langle \mathbf{c} \rangle}) = -w(\gamma_{kj}^{\psi a}, \pi_{k\langle \mathbf{c} \rangle}) = -w(\lambda_{kj}^{\psi a}, \pi_{k\langle \mathbf{c} \rangle}) = -w(\gamma_{kj}^{\varphi a}, \varrho_{k\langle \mathbf{c} \rangle}) = -w(\lambda_{kj}^{\varphi a}, \varrho_{k\langle \mathbf{c} \rangle}) = w(\gamma_{kj}^{\psi a}, \varrho_{k\langle \mathbf{c} \rangle}) = w(\lambda_{kj}^{\psi a}, \varrho_{k\langle \mathbf{c} \rangle}) = ([j]^{p_1})_{\langle \mathbf{c} \rangle}$ for $a \in \{0,1\}$, $j = 0, \ldots, 2^{p_1}-1$, and $w(z_i, \pi_{k\langle \mathbf{c} \rangle}) = w(z_i, \varrho_{k\langle \mathbf{c} \rangle}) = 2c_i - 1$ for $i = 1, \ldots, p_3$, and threshold $h(\pi_{k\langle \mathbf{c} \rangle}) = h(\varrho_{k\langle \mathbf{c} \rangle}) = 1 + \sum_{i=1}^{p_3} c_i$. Hence, neuron $\pi_{k\langle \mathbf{c} \rangle}$ is active iff $([\varphi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle} = 1$ and $([\psi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle} = 0$ for $y_{\text{in}} = a$, and $y_{z_i} = c_i$ for $i = 1, \ldots, p_3$, since only one pair of neurons $\gamma_{kj_\varphi}^{\varphi a}, \lambda_{kj_\varphi}^{\varphi a}$ for $0 \le j_\varphi \le 2^{p_1}-1$ fires such that $j_\varphi = \varphi_k^a(\mathbf{u}', \mathbf{v})$ and only one pair of units $\gamma_{kj_\psi}^{\psi a}, \lambda_{kj_\psi}^{\psi a}$ for $0 \le j_\psi \le 2^{p_1}-1$ is active such that $j_\psi = \psi_k^a(\mathbf{u}', \mathbf{v})$, while the remaining units in v_{21} are passive after κ_{d+1}^a fires. Analogously, neuron $\varrho_{k\langle \mathbf{c} \rangle}$

fires iff $([\varphi_k^a(\mathbf{u}',\mathbf{v})]^{p_1})_{\langle\mathbf{c}\rangle} = 0$ and $([\psi_k^a(\mathbf{u}',\mathbf{v})]^{p_1})_{\langle\mathbf{c}\rangle} = 1$ for $y_{\mathrm{in}} = a$, and $y_{z_i} = c_i$ for $i = 1,\ldots,p_3$.

It follows that for any $1 \leq k \leq p$, at most one unit among $\pi_{k\langle\mathbf{c}\rangle}, \varrho_{k\langle\mathbf{c}\rangle} \in \nu_3$ for $\mathbf{c} \in \{0,1\}^{p_3}$ is active, which determines the value of $f_k(a,\mathbf{u}',\mathbf{v},\mathbf{z})$ for $y_{\mathrm{in}} = a$ according (9). Thus, a binary encoding $\mathbf{f}(a,\mathbf{u}',\mathbf{v},\mathbf{z})$ of the new state of automaton A is computed as disjunctions (9) for $k = 1,\ldots,p$ by units from $\nu_0 \setminus \{\mathrm{in}\}$ (which rewrite the old state of A) using the recurrent connections leading from neurons of ν_3. After re-indexing the units in layer $\nu_0 \setminus \{\mathrm{in}\} = \{1,\ldots,p\}$ properly, for each $k = 1,\ldots,p$, we define weights $w(\pi_{k\langle\mathbf{c}\rangle},k) = w(\varrho_{k\langle\mathbf{c}\rangle},k) = 1$ for every $\mathbf{c} \in \{0,1\}^{p_3}$, and threshold $h(k) = 1$.

Now we specify the computational dynamics of neural network N simulating the finite automaton A. At the beginning, the states of neurons from $\nu_0 \setminus \{\mathrm{in}\}$ are placed in an initial state of A. Each bit x_i $(1 \leq i \leq n)$ of input word $\mathbf{x} = x_1,\ldots,x_n$, which is read by input neuron in $\in \nu_0$ at time instant $\tau(i-1)$ (i.e. $y_{\mathrm{in}}^{(\tau(i-1))} = x_i$), is being processed by N within the desired period of $\tau = d+4 = O(p2^{p_1}/e) = O(\sqrt{2^p}/e) = O(\sqrt{m}/e)$ time steps. The states of neurons in N are successively updated in the order following the architecture of layers. Thus, we define sets α_t of units updated at time instants $t \geq 1$ as $\alpha_{\tau(i-1)+1} = \nu_1$, $\alpha_{\tau(i-1)+j+1} = \nu_{12} \cup \nu_2$ for $j = 1,\ldots,d+1$, $\alpha_{\tau(i-1)+d+3} = \nu_{12} \cup \nu_2 \cup \nu_3$, and $\alpha_{\tau i} = \nu_0 \setminus \{\mathrm{in}\}$, for $i = 1,\ldots,n$. Eventually, the output neuron out $\in \nu_0$ signals at time instant τn whether input word $\mathbf{x}$ belongs to underlying language L, that is, $y_{\mathrm{out}}^{(\tau n)} = 1$ iff $\mathbf{x} \in L$.

The size of N simulating the finite automaton A with m states can be expressed as $s = |\nu_0| + |\nu_1| + |\nu_2| + |\nu_3| = p+1+2^{p_2}+2+8p2^{p_1}+2(d+1)+p2^{p_3} = O(\sqrt{2^p}) = O(\sqrt{m})$ in terms of m, which matches the known lower bound [2, 3]. Finally, the energy consumption can be bounded for particular layers as follows. Layer ν_0 can possibly require all $p+1$ units to fire for storing the binary encoding of a current automaton state. Moreover, there is only one active unit among neurons in ν_{11} which serve for evaluating all possible monomials over input variables $\mathbf{v}$, and also only one control unit from $\nu_{12} \cup \nu_{22}$ fires at one time instant. In addition, we know that the energy consumption by ν_{21} is at most $e+2$, and at most p neurons among $\pi_{k\langle\mathbf{c}\rangle}, \varrho_{k\langle\mathbf{c}\rangle}$ from ν_3 fire (one for each $k = 1,\ldots,p$). Altogether, the global energy consumption of N is bounded by $e+2p+5 = O(e+\log s) = O(e)$ as $e = \Omega(\log s)$ is assumed. This completes the proof of the theorem. □

5 Conclusions

We have, for the first time, applied the energy complexity measure to recurrent neural nets. This measure has recently been introduced and studied for feedforward perceptron networks. The binary-state recurrent neural networks recognize exactly the regular languages so we have investigated their energy consumption of simulating the finite automata with the asymptotically optimal number of neurons. We have presented a low-energy implementation of finite automata by optimal-size neural nets with the tradeoff between the time overhead for processing one input bit and the energy varying from the logarithm to the full network size. In the full paper, we will also present lower bounds on the energy complexity of neural network automata. In particular, for time overhead $\tau = O(1)$, the energy satisfies $e \geq s^{\varepsilon}$ for some real constant ε such that $0 < \varepsilon < 1$, and for infinitely many s, while for $\tau = O(\log^{\varepsilon} s)$, we have shown that $e = \Omega_{\infty}(s^{\log\log s / \log^{\eta} s})$ for any $\eta > \varepsilon$. An open problem remains for further research whether these bounds can be improved.

References

1. Alon, N., Dewdney, A.K., Ott, T.J.: Efficient simulation of finite automata by neural nets. Journal of the ACM 14(2), 495–514 (1991)
2. Horne, B.G., Hush, D.R.: Bounds on the complexity of recurrent neural network implementations of finite state machines. Neural Networks 9(2), 243–252 (1996)
3. Indyk, P.: Optimal simulation of automata by neural nets. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 337–348. Springer, Heidelberg (1995)
4. Lennie, P.: The cost of cortical computation. Current Biology 13(6), 493–497 (2003)
5. Lupanov, O.: On the synthesis of threshold circuits. Problemy Kibernetiki 26, 109–140 (1973)
6. Minsky, M.: Computations: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs (1967)
7. Siegelmann, H.T., Sontag, E.D.: Computational power of neural networks. Journal of Computer System Science 50(1), 132–150 (1995)
8. Šíma, J., Orponen, P.: General-purpose computation with neural networks: A survey of complexity theoretic results. Neural Computation 15(12), 2727–2778 (2003)
9. Šíma, J., Wiedermann, J.: Theory of neuromata. Journal of the ACM 45(1), 155–178 (1998)
10. Suzuki, A., Uchizawa, K., Zhou, X.: Energy and fan-in of threshold circuits computing Mod functions. In: Ogihara, M., Tarui, J. (eds.) TAMC 2011. LNCS, vol. 6648, pp. 154–163. Springer, Heidelberg (2011)
11. Uchizawa, K., Douglas, R., Maass, W.: On the computational power of threshold circuits with sparse activity. Neural Computation 18(12), 2994–3008 (2006)
12. Uchizawa, K., Nishizeki, T., Takimoto, E.: Energy and depth of threshold circuits. Theoretical Computer Science 411(44-46), 3938–3946 (2010)
13. Uchizawa, K., Takimoto, E.: Exponential lower bounds on the size of constant-depth threshold circuits with small energy complexity. Theoretical Computer Science 407(1-3), 474–487 (2008)
14. Uchizawa, K., Takimoto, E.: Lower bounds for linear decision trees via an energy complexity argument. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 568–579. Springer, Heidelberg (2011)
15. Uchizawa, K., Takimoto, E., Nishizeki, T.: Size-energy tradeoffs for unate circuits computing symmetric Boolean functions. Theoretical Computer Science 412(8-10), 773–782 (2011)

A Distributed Learning Algorithm Based on Frontier Vector Quantization and Information Theory

Diego Peteiro-Barral[⋆] and Bertha Guijarro-Berdiñas

Dept. of Computer Science, University of A Coruña,
Campus de Elviña s/n, 15071 A Coruñn, Spain
{dpeteiro,cibertha}@udc.es
http://www.lidiagroup.com

Abstract. In this paper, we propose a novel distributed learning algorithm built upon the Frontier Vector Quantization based on Information Theory (FVQIT) method. The FVQIT is very effective in classification problems but it shows poor training time performance. Thus, distributed learning is appropriate here to speed up training. One of the most promising lines of research towards learning from distributed data sets is separated learning and model integration. Separated learning avoids moving raw data around the distributed nodes. The integration of local models is implemented in this research using a genetic algorithm. The results obtained from twelve classification data sets demonstrate the efficacy of the proposed method. In average, the distributed FVQIT performs 13.56 times faster than the FVQIT and improves classification accuracy by 5.25%.

Keywords: Machine learning, distributed learning, FVQIT, neural networks, genetic algorithms.

1 Introduction

Distributed learning has become in a short time a fascinating area for many researchers in machine learning to address scalability of algorithms. Data sets have grown from few thousands of samples to several millions. In recent years, learning is limited by computational resources rather than availability of data. The current trend of reducing speed of processors in favor of multi-core processors and computer clusters leads to a suitable context for implementing distributed learning algorithms. The use of distributed partitions of data in learning may be significantly faster than traditional, batch learning.

In this research, we plan to distribute a classification method called FVQIT. The FVQIT uses techniques based on Information Theoretic Learning (ITL) [1]. ITL is a framework able to model information extracted from data through the expression of classical information theory concepts such as divergence, mutual information, etc. in

⋆ This research has been economically supported in part by the Secretaría de Estado de Investigación of the Spanish Government through the research projects TIN2009-10748 and TIN 2012-37954; and by the Consellería de Industria of the Xunta de Galicia through the research projects CN2011/007 and CN2012/211; all of them partially funded by FEDER funds of the European Union. Diego Peteiro-Barral acknowledges the support of Xunta de Galicia under *Plan I2C* Grant Program.

© Springer-Verlag Berlin Heidelberg 2013

terms of Renyis entropy and nonparametric probability density function estimators. The goal of the FVQIT [2] is to place a set of process elements (PEs) on the frontier between the classes, in such a way that each PE will represent a local model (see Figure 1). The philosophy of local models consists of splitting up the input space in several subspaces and adjusting a local model for each of these subspaces [3]. Each sub-problem is supposed to be simpler than the original and may be solved with simpler classification models. Thus, the FVQIT builds a piecewise representation of the borderline between classes in a classification problem. The FVQIT algorithm has proven to be effective in classification [2,4,5] in comparison with other methods such as SVM. However, the time the algorithm takes to train grows exponentially with respect to the number of input samples and PEs. This makes the FVQIT unsuitable for large data sets, and/or complex data sets in which a large number of PEs is required.

In this context, we can take advantage of the distributed paradigm to speed up the training process of the FVQIT. One of the most promising lines of research towards learning from distributed data sets is separated learning and model integration (Map/Reduce paradigm [6]). In the first stage, learners are trained on independent subsets of data (ideally, in parallel). Then, in the second stage, learners are integrated in some manner. Separated learning and model integration is appropriate in the case of the FVQIT. It is much faster on smaller training sets with smaller number of PEs. Moreover, separated learning avoids moving raw data around the distributed nodes, preserves privacy, and minimizes communication cost. Finally, the integration of local models is not a straightforward process. In this research, we propose to use a genetic algorithm (GA) to optimize the integration of local models. In the next section, local learning and model integration will be discussed in depth.

2 Distributed FVQIT (DFVQIT)

In this formulation, we assume that data is scattered across several nodes. In the first stage, an instance of the FVQIT algorithm is trained in every node.

2.1 Separated Learning

The FVQIT minimizes the energy function that calculates the divergence between the Parzen estimator of the distribution of data samples and the estimator of the distribution of the PEs. In this context, the Parzen density estimators of the distribution of data samples $f(\mathbf{x})$ and PEs $g(\mathbf{x})$ are:

$$\begin{aligned} f(\mathbf{x}) &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} G\left(\mathbf{x}-\mathbf{x}_i,\sigma_f^2\right) \\ g(\mathbf{x}) &= \tfrac{1}{M}\textstyle\sum_{j=1}^{M} G\left(\mathbf{x}-\mathbf{c}_j,\sigma_g^2\right) \end{aligned} \tag{1}$$

where N is the number of data samples, M is the number of PEs, G is a Gaussian kernel, σ_f^2 and σ_g^2 are the variances of the kernel functions, $\mathbf{x} \in \Re^n$ are data samples, and $\mathbf{c}_j \in \Re^n$ are the positions associated to the PEs. The function of energy $J(\mathbf{c})$ that calculates the divergence between the Parzen estimators is:

$$\begin{aligned} J(\mathbf{c}) = \log\int f^2(\mathbf{x})\,dx + 2\log\int f^+(\mathbf{x})\,g(\mathbf{x})\,dx - \\ -2\log\int f^-(\mathbf{x})\,g(\mathbf{x})\,dx + \log\int g^2(\mathbf{x})\,dx \end{aligned} \tag{2}$$

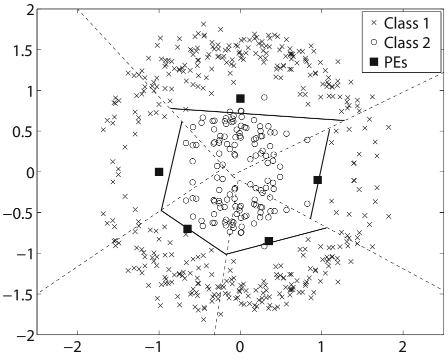


Fig. 1. Example of operation of FVQIT. Local models and frontier between classes. The division in local models is shown with dotted lines and the solid lines depict the decision regions.

where $f^{+}(\mathbf{x})$ and $f^{-}(\mathbf{x})$ are the estimators of the distributions of data for each of the two classes. In the multi-class version, the two nearest classes are chosen. According to [7], the first term of Eq. (2) is the information potential among data. For stationary data, this term will be zero. The second and third terms are the crossed correlations between the distributions of data and nodes. Finally, the fourth term is the information potential of the nodes. Assuming this formulation, when the PEs are placed on the minimum of the energy function $J(\mathbf{w})$, they are situated on a frontier area. Therefore, we utilize the gradient descent method to move the PEs toward such situation. The algorithm is summarized as follows

1. Initialize the positions $\mathbf{c}_i, i = 1, \ldots, M$ of the PEs randomly.
2. Compute the Euclidean distance from each PE to every data sample. Then, compute which class repels the PE and which class attracts it using the k-nearest neighbor rule [8]. For each PE $\mathbf{c}_i$,
   (a) Sort the data samples by increasing Euclidean distance to the PE.
   (b) Take the classes of the k closest points and calculate its mode. The mode will be the repelling class $(+)$ for that PE.
   (c) Take the classes of the k closest points to each PE that do not belong to the repelling class and calculate its mode. The mode will be the attracting class $(-)$ for that PE.
3. Compute the cross information potential C_+ between each PE and the samples from the repelling class, the cross information potential C_- between each PE and the data from the attracting class, and the entropy V between PEs. A physical interpretation of this premise leads to consider both data samples and PEs two kinds of particles with a potential field associated. These fields induce repulsive and attractive forces between particles. Additionally, a third force of repulsion between the PEs favors a better distribution, avoiding the accumulation of PEs on a point.
4. Update the positions of each PE through gradient descent,

$$\mathbf{c}_i(n+1) = \mathbf{c}_i(n) - \eta\left(\frac{\nabla V}{V} + \frac{\nabla C_+}{C_+} - \frac{\nabla C_-}{C_-}\right), i = 1, \ldots, M \tag{3}$$

   where n is the iteration and η is the step size.

5. Reduce the learning rate η by annealing rate η_{dec}, and σ_f and σ_g by σ_{dec}.
6. Repeat from 2 until the maximum number of iterations is reached.

Ideally, the PEs will find themselves well distributed on the frontiers between classes. Figure 1 shows a simple two-class bi-dimensional example for illustrative purposes. Each PE defines a region in the feature space. Those models are defined by proximity: the local model associated to each PE is composed of the nearest samples according to Euclidean distance in the feature space. At this moment, the goal is to construct a classifier for each local model. The FVQIT utilizes one-layer neural networks, a lightweight classifier trained with the efficient algorithm proposed in [9]. This algorithm allows rapid supervised training and requires less computational resources than classic methods. This classifier will be in charge of classifying samples in the region assigned to its local model and will be trained only with the points of the training set in this region (see Figure 1).

2.2 Model Integration

The integration of the distributed, partial solutions of the FVQIT in a comprehensive, global solution is not a straightforward process. A simple idea may be to combine every PE from every node but this approximation may lead to redundant, crowded areas in the input space. Figure 2 shows an example of the combination of the PEs from two different nodes. As can be seen, especially in the bottom part several PEs are too close one each other. This will affect the predictive ability of the model in the presence of noise.

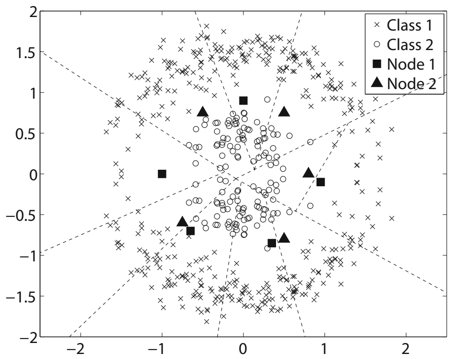


Fig. 2. Example of integration of two nodes of DFVQIT

In order to overcome this difficulty, we plan to exploit the advantages of the learning algorithm proposed in [9]. The key idea of the algorithm is to measure the error prior to the nonlinear activation functions. In this manner, the weights $\mathbf{w}$ of the ANN can be easily computed by solving a system of linear equations,

$$\mathbf{A} \cdot \mathbf{w} = \mathbf{b} \tag{4}$$

where

$$\begin{aligned} \mathbf{A} &= \mathbf{x} \cdot \mathbf{x}^T \\ \mathbf{b} &= f^{-1}(\mathbf{d}) \cdot \mathbf{x} \end{aligned} \tag{5}$$

x being the input samples, d the desired outputs, and f the nonlinear activation function. The advantage of this method is its ability to learn in an incremental fashion. Note that the solution to the previous equations is computed in terms of a sum. Thus, if we assume that data set $1 \cup 2$ contains the union of data sets 1 and 2, the following relation holds

$$\mathbf{A}^{1\cup 2} \cdot \mathbf{w} = \mathbf{b}^{1\cup 2} \Leftrightarrow (\mathbf{A}_1 + \mathbf{A_2}) \cdot \mathbf{w} = \mathbf{b}_1 + \mathbf{b}_2 \tag{6}$$

Based on this relation, we propose to implement a genetic algorithm (GA) [10] as integration method. The hypothesis is that the GA will optimize the initial population by pruning the crowded areas in the input space. Moreover, it is expected that including some global knowledge by means of a GA, will expand the search space and obtain a simpler and more general classifier. The challenge here is the encapsulation of a candidate solution. The piecewise representation of the FVQIT leads to complex crossover operators between nodes. Thus, we turn to consider each PE as an individual of the population in order to have a simple yet powerful method. This change in approach will cause that the optimization process will be driven by fitness-maximization of the population rather than best individual. Under these conditions, the GA we propose in this research will be implemented as follows. Note that in the FVQIT, each individual (PE) is defined by its location in the input space and its decision boundary.

- The fitness function will become the class accuracy of the entire population because of the piecewise representation of the FVQIT.
- Selection is the step in which individuals are chosen from the population for breeding. Note that individuals are usually selected based on their fitness but, in this implementation, the fitness function comprehend the entire population. However, we can take advantage of the locality of the FVQIT in order to propose an effectual selection method. In this research, the population is selected using roulette-wheel selection [10] by pairwise Euclidean distance between PEs; the closer the PEs the larger the probability of being selected for breeding.
- The crossover operator defines how two individuals of the population are combined together to produce the offspring. The crossover operator is computed as follows: the two parents generate a child which its location is the midpoint of their locations $\mathbf{c}^{child} = \left(\mathbf{c}^{parent_1} + \mathbf{c}^{parent_2}\right)/2$ and its decision boundary is the sum of their matrices of coefficients $\mathbf{A}^{child} = \mathbf{A}^{parent_1} + \mathbf{A}^{parent_2}$, and $\mathbf{b}^{child} = \mathbf{b}^{parent_1} + \mathbf{b}^{parent_2}$ which exploits the incremental features of the algorithm described above. Due to the incremental nature of the crossover operator, the child is expected to substitute both parents if this improves the fitness function.
- The mutation operator defines how an individual is altered to produce a new individual. In this research, small random disturbances are used to change the location $\mathbf{c}$ and the matrices of coefficients $\mathbf{A}$ and $\mathbf{b}$ of the individuals.

3 Experimental Study

In this section, we compare the distributed method with a similar batch scheme that uses in a comprehensive manner the entire set of samples. The pruning method is evaluated in both original and distributed FVQIT algorithm.

3.1 Materials and Methods

The algorithms are evaluated to classify twelve data sets of diverse kinds of tasks. Table 1 summarizes the number of input features, samples, and output classes of the data sets. A more detailed description of the twelve data sets can be found in [11].

Table 1. Brief description of the data sets

Name	Feats	Samples	Class	Name	Feats	Samples	Class
Abalone	8	4,177	28	Magic	10	19,020	2
Adult	33	30,162	2	Mushroom	22	8,124	2
Chess	6	28,056	18	Nursery	16	12,960	5
Connect4	42	67,557	3	Poker	10	25,010	10
Forest	54	101,241	7	Shuttle	9	43,500	7
Letter	16	20,000	26	Waveform	21	5,000	3

In distributed scenarios, training data have been scattered across 5 different nodes in which each node contains 10 PEs. In batch scenarios, the learning algorithm uses the entire data set in which the number of PEs is set to 50. The evaluation of the methods has been done using holdout, 90% for training, 10% for testing. When pruning is enabled, 10% of the training data have been used for validation. Experiments were run 10 times with random partitions of the data set. We use the Kruskal-Wallis test to check if there are significant differences, then we apply a multiple comparison procedure to find the methods which are not significantly different.

3.2 Results and Discussion

Table 2 shows the training time of the four implementations. The best result, or those not significantly different from the best one, are underlined for each data set. In average, DFVQIT performs 13.56 times faster than FVQIT. Furthermore, the larger the data set the larger the difference. In Connect4 and Forest data sets, DFVQIT is 19.24 and 22.72 times faster. If the genetic algorithm is used, pruned DFVQIT trains 4.26 and 4.63 times faster than the original FVQIT and the pruned FVQIT, respectively.

Table 3 presents the number of PEs at the end of training. In average, only the 36.63% and the 20.51% of the PEs are retained after pruning in the original FVQIT and DFVQIT, respectively. Note that the FVQIT algorithm is parametrized with the maximum number of PEs. If a PE do not cover any training sample, it will be deleted. It is important to remark that a smaller number of PEs is related with better generalization performance and faster execution.

Table 2. Trainnig time (s)

Data set	Original FVQIT	Pruned FVQIT	DFVQIT	Pruned DFVQIT
Abalone	9.82 ± 0.08	13.77 ± 0.82	1.10 ± 0.00	6.41 ± 0.51
Adult	96.06 ± 1.16	103.61 ± 2.56	6.33 ± 0.38	11.58 ± 2.00
Chess	24.39 ± 0.12	44.60 ± 1.21	2.15 ± 0.01	17.23 ± 1.61
Connect4	275.92 ± 5.08	294.09 ± 7.85	14.34 ± 0.10	33.80 ± 8.66
Forest	388.78 ± 9.41	517.17 ± 9.42	17.11 ± 0.40	142.67 ± 8.97
Letter	25.81 ± 0.13	57.88 ± 0.57	2.13 ± 0.01	32.41 ± 1.49
Magic	33.24 ± 2.32	41.68 ± 2.21	2.62 ± 0.17	7.51 ± 0.75
Mushroom	37.44 ± 3.16	32.80 ± 3.12	3.90 ± 0.30	7.63 ± 0.37
Nursery	25.96 ± 0.68	30.13 ± 1.51	2.13 ± 0.08	7.69 ± 0.76
Poker	38.06 ± 0.52	34.28 ± 2.00	3.15 ± 0.19	5.43 ± 2.11
Shuttle	52.60 ± 0.84	55.94 ± 1.38	3.17 ± 0.05	15.89 ± 0.69
Waveform	27.42 ± 3.77	26.93 ± 2.51	2.72 ± 0.48	5.27 ± 0.58

Table 3. Number of PEs

Data set	Original FVQIT	Pruned FVQIT	DFVQIT	Pruned DFVQIT
Abalone	47.10 ± 0.99	7.60 ± 2.22	49.90 ± 0.32	7.00 ± 1.70
Adult	49.90 ± 0.32	4.10 ± 2.47	50.00 ± 0.00	2.40 ± 1.35
Chess	49.90 ± 0.32	34.50 ± 2.92	50.00 ± 0.00	13.30 ± 2.54
Connect4	50.00 ± 0.00	6.70 ± 2.06	50.00 ± 0.00	2.40 ± 1.65
Forest	50.00 ± 0.00	35.89 ± 2.62	50.00 ± 0.00	17.89 ± 1.90
Letter	50.00 ± 0.00	48.40 ± 0.70	50.00 ± 0.00	27.40 ± 2.99
Magic	49.90 ± 0.32	9.80 ± 2.39	49.90 ± 0.32	7.10 ± 1.66
Mushroom	34.20 ± 2.70	13.50 ± 2.76	46.10 ± 1.45	11.30 ± 1.25
Nursery	41.60 ± 0.52	16.80 ± 3.39	48.40 ± 0.70	9.80 ± 2.44
Poker	43.80 ± 1.81	1.60 ± 0.70	50.00 ± 0.00	1.50 ± 0.71
Shuttle	44.20 ± 1.48	21.40 ± 2.59	49.00 ± 0.67	15.10 ± 2.56
Waveform	49.80 ± 0.42	6.21 ± 3.26	49.80 ± 0.42	6.20 ± 1.32

Table 4. Test error (%)

Data set	Original FVQIT	Pruned FVQIT	DFVQIT	Pruned DFVQIT
Abalone	84.00 ± 2.01	79.23 ± 2.49	82.37 ± 1.24	77.06 ± 2.58
Adult	24.24 ± 1.06	17.03 ± 0.66	25.18 ± 1.39	17.25 ± 0.83
Chess	62.95 ± 1.01	64.68 ± 1.51	73.27 ± 1.43	68.39 ± 1.17
Connect4	30.65 ± 0.41	25.99 ± 0.64	32.23 ± 0.79	25.96 ± 0.82
Forest	37.89 ± 1.27	36.31 ± 1.24	37.64 ± 1.14	34.87 ± 0.75
Letter	13.14 ± 1.01	14.31 ± 0.67	22.45 ± 0.96	22.84 ± 1.16
Magic	20.94 ± 0.83	15.79 ± 0.70	19.79 ± 1.65	15.69 ± 0.83
Mushroom	17.14 ± 2.29	11.53 ± 1.22	16.15 ± 2.57	11.34 ± 0.78
Nursery	7.24 ± 0.89	7.54 ± 1.09	8.74 ± 0.90	8.31 ± 0.87
Poker	72.74 ± 0.90	50.26 ± 1.01	72.71 ± 0.91	51.60 ± 1.13
Shuttle	0.51 ± 0.30	0.32 ± 0.13	1.92 ± 1.35	0.54 ± 0.12
Waveform	20.88 ± 2.09	16.14 ± 2.18	18.10 ± 1.86	17.16 ± 1.73

Finally, Table 4 shows the test error. The pruned algorithms obtain the best result, or not significantly different from the best one, in 10 out of 12 data sets (when compared to DFVQIT) and 12 out of 12 (when compared to FVQIT). The original FVQIT is only competitive with the proposed algorithms in 4 out of 12 data sets. In average, pruning improves classification accuracy by 5.43% and 5.96% with respect to the original FVQIT and plain DFVQIT, respectively.

4 Conclusions

In this research, we proposed a distributed algorithm DFVQIT that exploits the separated learning and model integration paradigm. If the time complexity of a learning algorithm is worse than linear, as is the case of the FVQIT, then processing smaller subsets of data concurrently can make it linear. Experimental results have demonstrated that the DFVQIT outperforms the original FVQIT not just in terms of training time, but also in the number of PEs and test accuracy. Moreover, the pruning method implemented has been proven effective in both FVQIT and DFVQIT.

For future work, we plan to extend this research to a more realistic scenario: distributions with data skew. Moreover, the evaluation of the effectiveness of several distributed algorithms will be evaluated in terms of scalability. Finally, it is an open question if it is worth it to use higher complexity classifiers as local models.

References

1. Principe, J.C.: Information theoretic learning: Renyi's entropy and kernel perspectives. Springer (2010)
2. Porto-Díaz, I., Martínez-Rego, D., Alonso-Betanzos, A., Fontenla-Romero, O.: Information theoretic learning and local modeling for binary and multiclass classification. In: Progress in Artificial Intelligence, pp. 1–14 (2012)
3. Dasarathy, B.V., Sheela, B.V.: A composite classifier system design: concepts and methodology. Proceedings of the IEEE 67(5), 708–713 (1979)
4. Martínez-Rego, D., Fontenla-Romero, O., Porto-Díaz, I., Alonso-Betanzos, A.: A new supervised local modelling classifier based on information theory. In: International Joint Conference on Neural Networks (IJCNN), pp. 2014–2020 (2009)
5. Porto-Dıaz, I., Alonso-Betanzos, A., Fontenla-Romero, O.: A multiclass classifier based on local modeling and information theoretic learning. In: Conference of the Spanish Association for Artificial Intelligence (CAEPIA) (2011)
6. Chu, C., Kim, S.K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. In: Advances in Neural Information Processing Systems, vol. 19, p. 281 (2007)
7. Principe, J.C., Xu, D., Zhao, Q., Fisher, J.W.: Learning from Examples with Information Theoretic Criteria. The Journal of VLSI Signal Processing 26(1), 61–77 (2000)
8. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
9. 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)
10. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning (1989)
11. Frank, A., Asuncion, A.: UCI machine learning repository (2010)

Efficient Baseline-Free Sampling in Parameter Exploring Policy Gradients: Super Symmetric PGPE

Frank Sehnke

Zentrum für Sonnenenergie- und Wasserstoff-Forschung,
Industriestr. 6, Stuttgart, BW 70565 Germany

Abstract. Policy Gradient methods that explore directly in parameter space are among the most effective and robust direct policy search methods and have drawn a lot of attention lately. The basic method from this field, Policy Gradients with Parameter-based Exploration, uses two samples that are symmetric around the current hypothesis to circumvent misleading reward in *asymmetrical* reward distributed problems gathered with the usual baseline approach. The exploration parameters are still updated by a baseline approach - leaving the exploration prone to asymmetric reward distributions. In this paper we will show how the exploration parameters can be sampled quasi symmetric despite having limited instead of free parameters for exploration. We give a transformation approximation to get quasi symmetric samples with respect to the exploration without changing the overall sampling distribution. Finally we will demonstrate that sampling symmetrically also for the exploration parameters is superior in needs of samples and robustness than the original sampling approach.

1 Introduction

Policy Gradient (PG) methods that explore directly in parameter space have some major advantages over standard PG methods, like described in [1,2,3,4,5,6] and [7] and have therefore drawn a lot of attention in the last years. The basic method from the field of Parameter Exploring Policy Gradients (PEPG) [8], Policy Gradients with Parameter-based Exploration (PGPE) [1], uses two samples that are symmetric around the current hypothesis to circumvent misleading reward in *asymmetrical* reward distributed problems, gathered with the usual baseline approach. [4] shows that Symmetric Sampling (SyS) is superior even to the optimal baseline. The exploration parameters, however, are still updated by a baseline approach - leaving the exploration prone to asymmetric reward distributions. While the optimal baseline improved this issue substantially, like shown again by [4], it is likely that removing the baseline altogether by a SyS wrt. the exploration parameters will be again superior. Because the exploration parameters are standard deviations that are bounded between zero and infinity, there exist no correct symmetric samples wrt. the exploration parameters.

© Springer-Verlag Berlin Heidelberg 2013

We will, however, show how the exploration parameters can be sampled quasi symmetric. We give therefore a transformation approximation to get quasi symmetric samples without changing the overall sampling distribution significantly, so that the PGPE assumptions based on normal distributed samples still hold. Finally we will demonstrate via experiments that sampling symmetrically also for the exploration parameters is superior in needs of samples and robustness compared to the original sampling approach, if confronted with search spaces with significant amounts of local optima.

2 Method

In this section we derive the super-symmetric sampling (SupSyS) method. We show how the method relates to SyS and sampling with a baseline, thereby summarizing the derivation from [1] for SyS and baseline sampling PGPE.

2.1 Parameter-Based Exploration

To stay conform with the nomenclature of [1] and [4], we assume a Markovian environment that produces a cumulative reward r for a fixed length *episode*, *history*, *trajectory* or *roll-out*. In this setting, the goal of reinforcement learning is to find the optimal policy parameters $\boldsymbol{\theta}$ that maximize the agent's expected reward

$$J(\boldsymbol{\theta}) = \int_H p(h|\boldsymbol{\theta})r(h)dh. \tag{1}$$

An obvious way to maximize $J(\boldsymbol{\theta})$ is to estimate $\nabla_{\boldsymbol{\theta}} J$ and use it to carry out gradient ascent optimization. The probabilistic policy used in standard PG is replaced with a probability distribution over the parameters $\boldsymbol{\theta}$ for PGPE. The advantage of this approach is that the actions are deterministic, and an entire history can therefore be generated from a single parameter sample. This reduction in samples-per-history is what reduces the variance in the gradient estimate (see [1] for details).

We name the distribution over parameters in accordance with [1] $\boldsymbol{\rho}$. The expected reward with a given $\boldsymbol{\rho}$ is

$$J(\boldsymbol{\rho}) = \int_{\boldsymbol{\Theta}} \int_H p(h, \boldsymbol{\theta}|\boldsymbol{\rho})r(h)dhd\boldsymbol{\theta}. \tag{2}$$

Differentiating this form of the expected return with respect to $\boldsymbol{\rho}$ and applying sampling methods (first choosing $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\boldsymbol{\rho})$, then running the agent to generate h from $p(h|\boldsymbol{\theta})$) yields the following gradient estimator:

$$\nabla_{\boldsymbol{\rho}} J(\boldsymbol{\rho}) \approx \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}|\boldsymbol{\rho})r(h^n). \tag{3}$$

Assuming that $\boldsymbol{\rho}$ consists of a set of means $\{\mu_i\}$ and standard deviations $\{\sigma_i\}$ that determine an independent normal distribution for each parameter θ_i in

$\boldsymbol{\theta}$ gives the following forms for the derivative of the characteristic eligibility $\log p(\boldsymbol{\theta}|\boldsymbol{\rho})$ with respect to μ_i and σ_i

$$\nabla_{\mu_i} \log p(\boldsymbol{\theta}|\boldsymbol{\rho}) = \frac{(\theta_i - \mu_i)}{\sigma_i^2}, \qquad \nabla_{\sigma_i} \log p(\boldsymbol{\theta}|\boldsymbol{\rho}) = \frac{(\theta_i - \mu_i)^2 - \sigma_i^2}{\sigma_i^3}, \tag{4}$$

which can be substituted into Eq. (3) to approximate the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ gradients.

2.2 Sampling with a Baseline

Given enough samples, Eq. (3) will determine the reward gradient to arbitrary accuracy. However each sample requires rolling out an entire state-action history, which is expensive. Following [9], we obtain a cheaper gradient estimate by drawing a single sample $\boldsymbol{\theta}$ and comparing its reward r to a baseline reward b given e.g. by a moving average over previous samples. Intuitively, if $r > b$ we adjust $\boldsymbol{\rho}$ so as to increase the probability of $\boldsymbol{\theta}$, and $r < b$ we do the opposite. If, as in [9], we use a step size $\alpha_i = \alpha\sigma_i^2$ in the direction of positive gradient (where α is a constant) we get the following parameter update equations:

$$\Delta\mu_i = \alpha(r-b)(\theta_i - \mu_i), \qquad \Delta\sigma_i = \alpha(r-b)\frac{(\theta_i - \mu_i)^2 - \sigma_i^2}{\sigma_i}. \tag{5}$$

Usually the baseline is realized as decaying or moving average baseline of the form:

$$b(n) = \gamma r(h^{n-1}) + (1-\gamma)b(n-1) \qquad \text{or} \qquad b(n) = \sum_{n=N-m}^{N} r(h^n)/m \tag{6}$$

[4] showed recently that an optimal baseline can be achieved for PGPE and the algorithm converges significantly faster with an optimal baseline of the form:

$$b^* = \frac{\mathbb{E}[r(h)||\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}|\boldsymbol{\rho})||^2]}{\mathbb{E}[||\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}|\boldsymbol{\rho})||^2]}. \tag{7}$$

2.3 Symmetric Sampling

While sampling with a baseline is efficient and reasonably accurate for most scenarios, it has several drawbacks. In particular, if the reward distribution is strongly skewed then the comparison between the sample reward and the baseline reward is misleading. A more robust gradient approximation can be found by measuring the difference in reward between two symmetric samples on either side of the current mean. That is, we pick a perturbation $\boldsymbol{\epsilon}$ from the distribution $\mathcal{N}(0, \boldsymbol{\sigma})$, then create symmetric parameter samples $\boldsymbol{\theta}^+ = \boldsymbol{\mu} + \boldsymbol{\epsilon}$ and $\boldsymbol{\theta}^- = \boldsymbol{\mu} - \boldsymbol{\epsilon}$. Defining r^+ as the reward given by $\boldsymbol{\theta}^+$ and r^- as the reward given by $\boldsymbol{\theta}^-$. We can insert the two samples into Eq. (3) and make use of Eq. (4) to obtain

$$\nabla_{\mu_i} J(\boldsymbol{\rho}) \approx \frac{\epsilon_i(r^+ - r^-)}{2\sigma_i^2}, \tag{8}$$

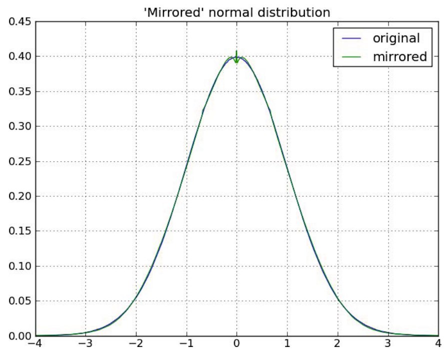


Fig. 1. Normal distribution and the final approximation of the 'mirrored' distribution

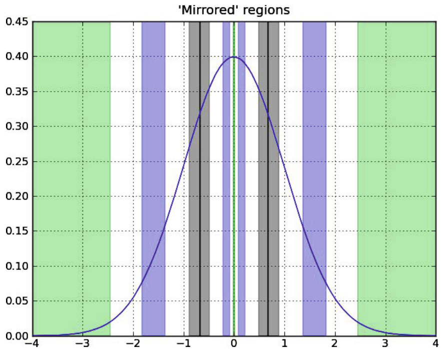


Fig. 2. Normal distribution and the regions that are transfered into each other by 'reflecting' the samples on the other side of the median deviation

which resembles the *central difference* approximation used in finite difference methods. Using the same step sizes as before gives the following update equation for the $\boldsymbol{\mu}$ terms

$$\Delta\mu_i = \frac{\alpha\epsilon_i(r^+ - r^-)}{2}. \tag{9}$$

The updates for the standard deviations are more involved. As $\boldsymbol{\theta}^+$ and $\boldsymbol{\theta}^-$ are by construction equally probable under a given $\boldsymbol{\sigma}$, the difference between them cannot be used to estimate the $\boldsymbol{\sigma}$ gradient. Instead we take the mean $\frac{r^+ + r^-}{2}$ of the two rewards and compare it to the baseline reward b. This approach yields

$$\Delta\sigma_i = \alpha\left(\frac{r^+ + r^-}{2} - b\right)\left(\frac{\epsilon_i^2 - \sigma_i^2}{\sigma_i}\right) \tag{10}$$

SyS removes the problem of misleading baselines, and therefore improves the $\boldsymbol{\mu}$ gradient estimates. It also improves the $\boldsymbol{\sigma}$ gradient estimates, since both samples are equally probable under the current distribution, and therefore reinforce each other as predictors of the benefits of altering $\boldsymbol{\sigma}$. Even though symmetric sampling requires twice as many histories per update, [1] and [4] have shown that it gives a considerable improvement in convergence quality and time.

2.4 Super-Symmetric Sampling

While SyS removes the misleading baseline problem for the $\boldsymbol{\mu}$ gradient estimate, the $\boldsymbol{\sigma}$ gradient still uses a baseline and is prone to this problem. On the other hand there is no correct *symmetric* sample with respect to the standard deviation, because the standard deviation is bounded on the one *side* to 0 and is unbounded on the positive *side*. Another problem is that $\frac{2}{3}$ of the samples are on one *side* of the standard deviation and only $\frac{1}{3}$ on the other - *mirroring* the samples to the opposite side of the standard deviation in some way, would therefore deform the normal distribution so much, that it would no longer be a close enough approximation to fulfill the assumptions that lead to the PGPE update rules.

We therefore chose to define the normal distribution via the mean and the median deviation $\boldsymbol{\phi}$. The median deviation is due to the nice properties of the normal distribution simply defined by: $\boldsymbol{\phi} = 0.67449 \cdot \boldsymbol{\sigma}$. We can therefore draw samples from the new defined normal distribution: $\boldsymbol{\epsilon} \sim \mathcal{N}_m(0, \boldsymbol{\phi})$.

The median deviation has by construction an equal amount of samples on either side and solves therefore the symmetry problem of *mirroring* samples. The update rule Eq. (9) stays unchanged while Eq. (10) is only scaled by $\frac{1}{0.67449}$ (the factor that transforms $\boldsymbol{\phi}$ in $\boldsymbol{\sigma}$) that can be substituted in $\alpha_{\boldsymbol{\sigma}}$.

While the update rules stay the same for normal distributed sampling using the median deviation (despite a larger $\alpha_{\boldsymbol{\sigma}}$), the median deviation is still also bounded on one side. Because the *mirroring* cannot be solved in closed form we resort to approximation via a polynomial that can be transfered to an infinite series. We found a good approximation for *mirroring* samples by:

$$a_i = \frac{\phi_i - \mid \epsilon_i \mid}{\phi_i}, \qquad \epsilon_i^* = sign(\epsilon_i) \cdot \phi_i \cdot \begin{cases} e^{c_1 \frac{|a_i|^3 - |a_i|}{\log(|a_i|)} + c_2 |a_i|} & \text{if } a_i \leq 0 \\ e^{a_i} / (1. - a_i^3)^{c_3 a_i} & \text{if } a_i > 0, \end{cases} \tag{11}$$

with the following constants: $c_1 = -0.06655, c_2 = -0.9706, c_3 = 0.124$. This *mirrored* distribution has a standard deviation of 1.002 times the original standard deviation and looks like depicted in Fig. 1. Fig. 2 shows the regions of samples that are transfered into each other while generating the quasi symmetric samples.

Additional to the symmetric sample with respect to the mean hypothesis, now we also can generate two quasi symmetric samples with respect to the median deviation. We named this set of four samples super symmetric samples (SupSyS-samples). They allow for completely baseline free update rules, not only for the $\boldsymbol{\mu}$ update but also for the $\boldsymbol{\sigma}$ updates.

Therefore the two symmetric sample pairs are used to update $\boldsymbol{\mu}$ according to Eq. (9). $\boldsymbol{\sigma}$ is updated in a similar way by using the mean reward of each symmetric sample pair, there r^{++} is the mean reward of the original symmetric sample pair and r^{--} is the mean reward of the *mirrored* sample pair. The SupSyS update rule for the $\boldsymbol{\sigma}$ update is given by:

$$\Delta\sigma_i = \frac{\alpha \frac{\epsilon_i^2 - \sigma_i^2}{\sigma_i} (r^{++} - r^{--})}{2}. \tag{12}$$

3 Experiments and Results

We use the square function as search space instance with no local optima and the Rastrigin function (see Fig. 8) as search space with exponentially many local optima, to test the different behavior of SupSyS- and SyS-PGPE. The two metaparameters connected with SyS-PGPE as well as with SupSyS-PGPE, namely

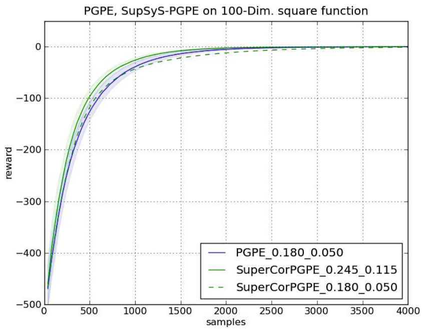


Fig. 3. Convergence plots of PGPE and SupSyS-PGPE on the 100 dimensional square function. The mean and standard deviation of 200 independent runs are shown.

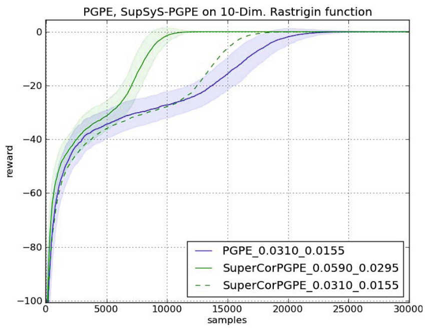


Fig. 4. Convergence plots of PGPE and SupSyS-PGPE on the 10 dimensional Rastrigin function. The mean and standard deviation of 200 independent runs are shown.

the step sizes for the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ updates, were optimized for every experiment via a grid search. The Figures 3 to 6 show the means and standard deviations of 200 independent runs each. It can be seen in Fig. 3 that for a search space with no local optima SupSyS-PGPE shows no advantage over standard SyS-PGPE. However, despite using 4 samples per update the performance is also not reduced by using SupSyS-PGPE — the two methods become merely equivalent. The situation changes drastically if the Rastrigin function is used as test function. Not only needs SupSyS-PGPE about half the samples compared to PGPE, the effect seems also to become stronger the higher dimensional the search space

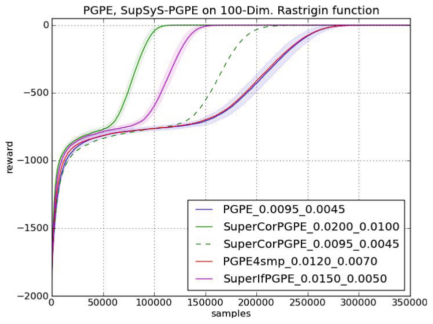


Fig. 5. Convergence plots of PGPE, PGPE with 4 samples (PGPE4smp), conditional SupSyS-PGPE (SupIf-PGPE) and SupSyS-PGPE on the 100 dimensional Rastrigin function. The mean and standard deviation of 200 independent runs are shown.

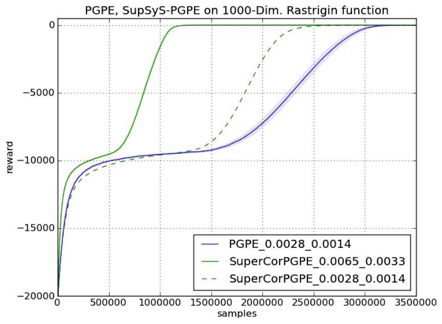


Fig. 6. Convergence plots of PGPE and SupSyS-PGPE on the 1000 dimensional Rastrigin function. The mean and standard deviation of 200 independent runs are shown.

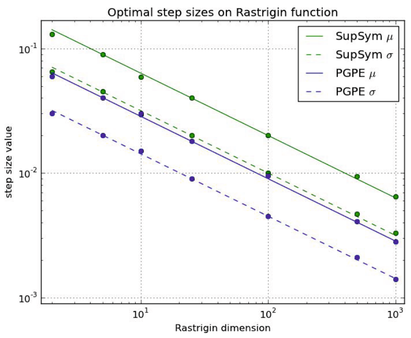


Fig. 7. Optimal meta-parameters for the multi-dimensional Rastrigin function for PGPE and SupSyS-PGPE

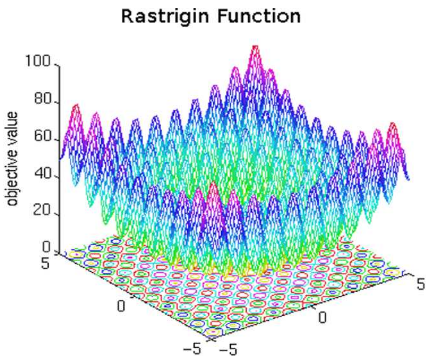


Fig. 8. Visualization of the 2D Rastrigin function

gets (see Fig. 4 to Fig. 6). We also added SupSyS-PGPE plots with the for SyS-PGPE optimal (less greedy) meta parameters to show that the effect is not only due to the more aggressive meta parameters. This runs were also more efficient than for PGPE, the effect was however not so distinct.

In Fig. 5 we also show a standard PGPE experiment with 4 samples (2 SyS samples — *PGPE4smp*) instead of 2 to show that the improved performance is not due to the different samples per update. Fig. 5 additionally shows an experiment (SupIf-PGPE) there symmetric samples are only drawn if the first sample(s) result in worse reward than a decaying average baseline. The intuitive idea behind symmetric samples was initially that changing the parameters *away* from the current sample if the sample resulted in lower than average reward may move the mean hypothesis still in a worse region of the parameter space. Search spaces like the one given in the Rastrigin function can visualize this problem. For SupIf-PGPE one Sample is drawn. If the reward is larger than the baseline then an update is done immediately. If not, a symmetric sample is drawn. Is the mean reward connected with both samples better than the baseline an SyS-PGPE update is done. If also this mean reward is worse than the baseline, a full SupSyS-PGPE update with 2 additional SyS samples is performed. As can be seen in Fig. 5 the performance is worse by some degree — the difference is however small enough that maybe the optimal baseline approach would improve this method enough to be challenging to SupSyS-PGPE (see also Sec. 4).

The optimal meta-parameters are an exponential function of the search space dimension, like to expect, so that we observe a line in the *loglog*-plot of Fig. 7. For SupSyS-PGPE the meta-parameters are about 2 times larger than for SyS-PGPE. This is partly because SupSyS-PGPE uses four samples per update instead of two. But the optimal meta-parameters are also larger than for the PGPE4smp experiment so that the symmetric nature of the four SupSyS samples obviously brings additional stability in the gradient estimate than a pure averaging over 4 samples would.

4 Conclusions and Future Work

We introduced SupSyS-PGPE, a completely baseline free PGPE that uses quasi-symmetric samples wrt. the exploration parameters. We showed that on the Rastrigin function, as example for a test function with exponentially many local optima, this novel method is clearly superior to standard SyS-PGPE and that both methods become equivalent in performance if the search space lack *distracting* local optima.

For future work we want to highlight that SupSyS-PGPE can be easily combined with other extensions of PGPE. Multi-modal PGPE [10] can be equipped straight forward with SupSyS sampling. Also the natural gradient used for PGPE in [3] can be defined over the SupSyS gradient instead over the vanilla gradient. If the full 4 super symmetric sample set is only used if the first samples are worse than a baseline (like described as SupIf-PGPE in Sec. 3) a combination with the optimal baseline (described for PGPE in [4]) can yield a superior method to both SupSyS-PGPE and optimal baseline PGPE. Also importance mixing introduced for PGPE by [5] is applicable to SupSyS-PGPE.

Finally a big open point for future work is the validation of the mere theoretical findings on real world problems, e.g. robotic tasks, for SupSyS-PGPE and its combination with other PGPE extensions.

References

1. Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., Schmidhuber, J.: Parameter-exploring policy gradients. Neural Networks 23(4), 551–559 (2010)
2. Rückstieß, T., Sehnke, F., Schaul, T., Wierstra, D., Sun, Y., Schmidhuber, J.: Exploring parameter space in reinforcement learning. Paladyn. Journal of Behavioral Robotics 1(1), 14–24 (2010)
3. Miyamae, A., Nagata, Y., Ono, I.: Natural Policy Gradient Methods with Parameter-based Exploration for Control Tasks. In: NIPS, pp. 1–9 (2010)
4. Zhao, T., Hachiya, H., Niu, G., Sugiyama, M.: Analysis and improvement of policy gradient estimation. Neural networks: the Official Journal of the International Neural Network Society, 1–30 (October 2011)
5. Zhao, T., Hachiya, H., Tangkaratt, V., Morimoto, J., Sugiyama, M.: Efficient sample reuse in policy gradients with parameter-based exploration. arXiv preprint arXiv:1301.3966 (2013)
6. Stulp, F., Sigaud, O.: Path integral policy improvement with covariance matrix adaptation. arXiv preprint arXiv:1206.4621 (2012)
7. Wierstra, D., Schaul, T., Peters, J., Schmidhuber, J.: Natural evolution strategies. In: Evolutionary Computation, CEC 2008, pp. 3381–3387. IEEE (2008)
8. Sehnke, F.: Parameter exploring policy gradients and their implications
9. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8, 229–256 (1992)
10. Sehnke, F., Graves, A., Osendorfer, C., Schmidhuber, J.: Multimodal parameter-exploring policy gradients. In: 2010 Ninth International Conference on Machine Learning and Applications (ICMLA), pp. 113–118. IEEE (2010)

Direct Method for Training Feed-Forward Neural Networks Using Batch Extended Kalman Filter for Multi-Step-Ahead Predictions

Artem Chernodub

Institute of Mathematical Machines and Systems NASU, Neurotechnologies Dept., Glushkova 42 ave., 03187 Kyiv, Ukraine
a.chernodub@gmail.com

Abstract. This paper is dedicated to the long-term, or multi-step-ahead, time series prediction problem. We propose a novel method for training feed-forward neural networks, such as multilayer perceptrons, with tapped delay lines. Special batch calculation of derivatives called Forecasted Propagation Through Time and batch modification of the Extended Kalman Filter are introduced. Experiments were carried out on well-known timeseries benchmarks, the Mackey-Glass chaotic process and the Santa Fe Laser Data Series. Recurrent and feed-forward neural networks were evaluated.

Keywords: multi-step-ahead prediction, Batch Extended Kalman Filter, Forecasted Propagation Through Time.

1 Introduction

Time series forecasting is a current scientific problem that has many applications in control theory, economics, medicine, physics and other domains. Neural networks are known as an effective and friendly tool for black-box modeling of plant's dynamics [1]. Usually, neural networks are trained to perform single-step-ahead (SS) predictions, where the predictor uses some available input and output observations to estimate the variable of interest for the time step immediately following the latest observation [2-4]. However, recently there has been growing interest in multi-step-ahead (MS) predictions, where the values of interest must be predicted for some horizon in the future. Knowing the sequence of future values allows for estimation of projected amplitudes, frequencies, and variability, which are important for modeling predictive control [5], flood forecasts [6] and fault diagnostics [7]. Generally speaking, the ability to perform MS predictions is frequently treated as the "true" test for the quality of a developed empirical model. In particular, well-known echo state machine neural networks (ESNs) became popular because of their ability to perform good long-horizon $(H = 84)$ multistep predictions [8].

The most straightforward approach to perform MS prediction is to train the SS predictor first and then use it in an autonomous "closed-loop" mode. The predictor's output is fed back to the input for a finite number of time steps. However, this simple

© Springer-Verlag Berlin Heidelberg 2013

method frequently shows poor results because of the accumulation of errors on difficult data points [4]. Recurrent neural networks (RNNs) such as NARX and Elman networks usually show better results. They are based on the calculation of special dynamic derivatives called Backpropagation Through Time (BPTT). The underlying idea of BPTT is to calculate derivatives by propagating the errors back across the RNN, which is unfolded through time. This penalizes the predictor for accumulating errors in time and therefore provides better MS predictions. Nonetheless, RNNs have some disadvantages. First, the implementation of RNNs is harder than feed-forward neural networks (FFNNs) in industrial settings. Second, training the RNNs is a difficult problem because of their more complicated error surfaces and vanishing gradient effects [9]. Third, the internal dynamics of RNNs make them less friendly for stability analysis. All of the above reasons prevent RNNs from becoming widely popular in industry. Meanwhile, RNNs have inspired a new family of methods for training FFNNs to perform MS predictions called direct methods [4]. Accumulated error is backpropagated through an unfolded through time FFNN in BPTT style that causes minimization of the MS prediction error. Nevertheless, the vanishing gradient effect still occurs in all multilayer perceptron-based networks with sigmoidal activation functions.

We propose a new, effective method for training the feed-forward neural models to perform MS prediction, called Forecasted Propagation Through Time (FPTT), for calculating the batch-like dynamic derivatives that minimize the negative effect of vanishing gradients. We use batch modification of the EKF algorithm which naturally deals with these batch-like dynamic derivatives for training the neural network.

2 Modeling Time Series Dynamics

We consider modeling time series in the sense of dealing with generalized nonlinear autoregression (NAR) models. In this case, time series behavior can be captured by expressing the observable value $y(k+1)$ as a function of N previous values $y(k),..., y(k-N+1)$:

$$y(k+1) = F(y(k), y(k-1),..., y(k-N+1)), \tag{1}$$

where k is the time step variable and $F(\cdot)$ is an unknown function that defines the underlying dynamic process. The goal of training the neural network is to develop the empirical model of function $F(\cdot)$ as closely as possible. If such a neural model $\tilde{F}(\cdot)$ is available, one can perform iterated multi-step-ahead prediction:

$$\tilde{y}(k+1) = \tilde{F}(y(k), y(k-1),..., y(k-N+1)), \tag{2}$$

$$\dots$$

$$\tilde{y}(k+H+1) = \tilde{F}(\tilde{y}(k+H), \tilde{y}(k+H-1),..., \tilde{y}(k+H-N+1)), \tag{3}$$

where $\tilde{y}$ is the neural network's output and H is the horizon of prediction.

2.1 Training Traditional Multilayer Perceptrons Using EKF for SS Predictions

Dynamic Multilayer Perceptron. Dynamic multilayer perceptrons (DMLP) are the most popular neural network architectures for time series prediction. Such neural networks consist of multilayer perceptrons with added tapped delay line of order N (Fig. 1, left).

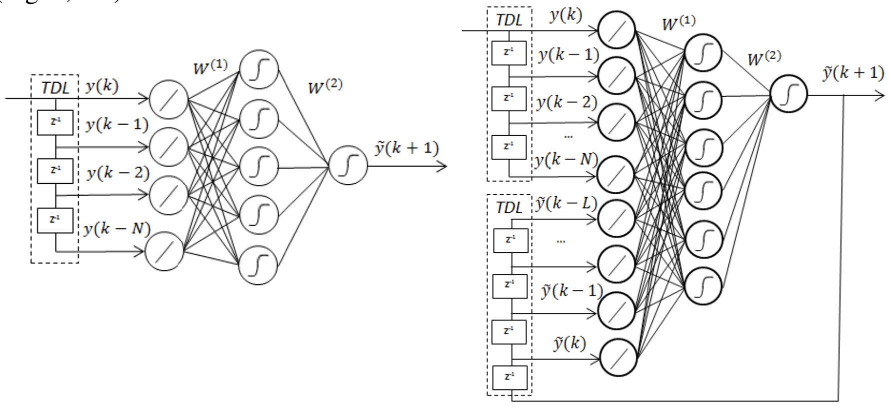


Fig. 1. DMLP neural network (left), NARX neural network (right)

The neural network receives an input vector $x(k) = [y(k) \quad y(k-1) \quad .... \quad y(k-N)]^T$, and calculates the output $\tilde{y}(k+1) = g(\sum_j w_j^{(2)}(f(\sum_i w_{ji}^{(1)} x_i)))$, where $w^{(1)}$ and $w^{(2)}$ are weights of the hidden and output layers and $f(\cdot)$ and $g(\cdot)$ are activation functions of the hidden and output layers.

Calculation of BP Derivatives. The Jacobians $\frac{\partial \tilde{y}}{\partial w}$ for the neural network's training procedure are calculated using a standard backpropagation technique by propagating a constant value $\delta^{OUT} = 1$ at each backward pass instead of propagating the residual error $\delta^{OUT} = t(k+1) - \tilde{y}(k+1)$ which calculates Jacobians $\frac{\partial \tilde{y}(k)}{\partial w}$ instead of error gradients $\frac{\partial E(k)}{\partial w}$ because $\frac{\partial E(k)}{\partial w} = \frac{\partial [e(k+1)^2]}{\partial w} = 2e(k+1)\frac{\partial y}{\partial w}$.

Extended Kalman Filter Method for Training DMLP. Although EKF training [7], [10] is usually associated with RNNs it can be applied to any differentiable parameterized model. Training the neural network using an Extended Kalman Filter may be considered a state estimation problem of some unknown "ideal" neural network that provides zero residual. In this case, the states are the neural network's weights $w(k)$ and the residual is the current training error $e(k+1) = t(k+1) - \tilde{y}(k+1)$. During the initialization step, covariance matrices of measurement noise $R = \eta I$ and dynamic training noise $Q = \mu I$ are set. Matrix R has size $L_w \times L_w$, matrix Q has size $N_w \times N_w$, where L_w is the number of output neurons, and N_w is the number of the network's weight coefficients. Coefficient η is the training speed, usually $\eta \sim 10^{-2}...10^{-4}$, and coefficient μ defines the measurement noise, usually $\mu \sim 10^{-4}...10^{-8}$. Also, the identity covariance matrix P of size $N_w \times N_w$ and zero observation matrix H of size $L_w \times N_w$ are defined. The following steps must be performed for all elements of the training dataset:

1) Forward pass: the neural network's output $\tilde{y}(k+1)$ is calculated.

2) Backward pass: Jacobians $\frac{\partial \tilde{y}}{\partial w}$ are calculated using backpropagation. Observation matrix $H(k)$ is filled:

$$H(k) = \begin{bmatrix} \frac{\partial \tilde{y}(k+1)}{\partial w_1} & \frac{\partial \tilde{y}(k+1)}{\partial w_2} & \cdots & \frac{\partial \tilde{y}(k+1)}{\partial w_{N_w}} \end{bmatrix}. \tag{4}$$

3) Residual matrix $E(k)$ is filled:

$$E(k) = [e(k+1)]. \tag{5}$$

4) New weights $w(k)$ and correlation matrix $P(k+1)$ are calculated:

$$K(k) = P(k)H(k)^T [H(k)P(k)H(k)^T + R]^{-1}, \tag{6}$$

$$P(k+1) = P(k) - K(k)H(k)P(k) + Q, \tag{7}$$

$$w(k+1) = w(k) + K(k)E(k). \tag{8}$$

2.2 Training NARX Networks Using BPTT and EKF

Nonlinear Autoregression with eXternal Inputs. The NARX neural network structure is shown in Fig. 1. It is equipped with both a tapped delay line at the input and global recurrent feedback connections, so the input vector $x(k) = [y(k) \ \ldots \ y(k-N) \ \tilde{y}(k) \ \ldots \ \tilde{y}(k-L)]^T$, where N is the order of the input tapped delay and L is the order of the feedback tapped delay line.

Calculation of BPTT Derivatives. Jacobians are calculated according to the BPTT scheme [1, p. 836], [4], [7]. After calculating the output $\tilde{y}(k+1)$, the NARX network is unfolded back through time. The recurrent neural network is presented as an FFNN with many layers, each corresponding to one retrospective time step $k-1$, $k-2$, ..., $k-h$, where h is a BPTT truncation depth. The set of static Jacobians $\frac{\partial \tilde{y}(k-n)}{\partial w}$ are calculated for each of the unrolled retrospective time steps. Finally, dynamic BPTT Jacobians $\frac{\partial \tilde{y}_{BPTT}(k)}{\partial w}$ are averaged static derivatives obtained for the feed-forward layers.

Extended Kalman Filter Method for Training NARX. Training the NARXs using an EKF algorithm is accomplished in the same way as training the DMLPs described above. The only difference is that the observation matrix H is filled by the dynamic derivatives $\frac{\partial \tilde{y}_{BPTT}(k)}{\partial w^{(1)}}$ and $\frac{\partial \tilde{y}_{BPTT}(k)}{\partial w^{(2)}}$, which contain temporal information.

2.3 Direct Method of Training Multilayer Perceptrons for MS Predictions Using FPTT and Batch EKF

Calculation of FPTT Derivatives. We propose a new batch-like method of calculating the dynamic derivatives for the FFNNs called Forecasted Propagation Through Time (Fig. 3).

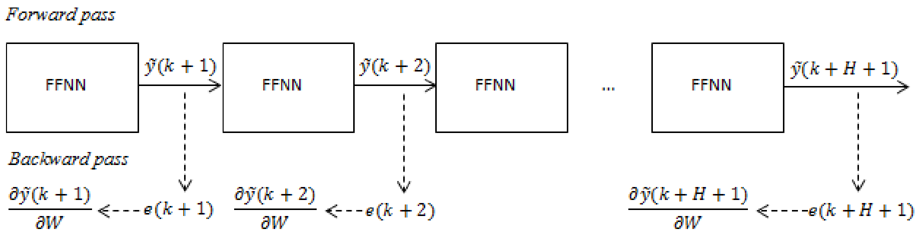

Forward pass
FFNN
$\tilde{y}(k+1)$
FFNN
$\tilde{y}(k+2)$
FFNN
...
FFNN
$\tilde{y}(k+H+1)$
Backward pass
$\frac{\partial \tilde{y}(k+1)}{\partial W}$
$e(k+1)$
$\frac{\partial \tilde{y}(k+2)}{\partial W}$
$e(k+2)$
$\frac{\partial \tilde{y}(k+H+1)}{\partial W}$
$e(k+H+1)$

Fig. 2. Calculation of dynamic FPTT derivatives for feedforward neural networks

1) At each time step the neural network is unfolded forward through time H times using Eqs. (2)-(3) in the same way as it is performed for regular multi-step-ahead prediction, where H is a horizon of prediction. Outputs $\tilde{y}(k+1),..., \tilde{y}(k+H+1)$ are calculated.
2) For each of H forecasted time steps, prediction errors $e(k+h+1)=t(k+h+1)-\tilde{y}(k+h+1)$, $h=1,...,H$ are calculated.
3) The set of independent derivatives $\left\{\frac{\partial \tilde{y}(k+h)}{\partial W}\right\}$, $h=1,...,H+1$, using the standard backpropagation of independent errors $\{e(k+h)\}$ are calculated for each copy of the unfolded neural network.

There are three main differences between the proposed FPTT and traditional BPTT. First, BPTT unfolds the neural network backward through time; FPTT unfolds the neural network recursively forward through time. This is useful from a technological point of view because this functionality must be implemented for MS predictions anyway. Second, FPTT does not backpropagate the accumulated error through the whole unfolded structure. It instead calculates BP for each copy of the neural network. Finally, FPTT does not average derivatives, it calculates a set of formally independent errors and a set of formally independent derivatives for future time steps instead. By doing this, we leave the question about contributions of each time step to the total MS error to the Batch EKF Algorithm.

Batch Extended Kalman Filter Method for Training DMLP Using FPTT. The EKF training algorithm also has a batch form [11]. In this case, a batch size of H patterns and a neural network with L_w outputs is treated as training a single shared-weight network with $L_w \times H$ outputs, i.e. H data streams which feed H networks constrained to have identical weights are formed from the training set. A single weight update is calculated and applied equally to each stream's network. This weights update is sub-optimal for all samples in the batch. If streams are taken from different places in the dataset, then this trick becomes equivalent to a Multistream EKF [7], [10], a well-known technique for avoiding poor local minima. However, we use it for direct minimization of accumulated error H steps ahead. Batch observation matrix $\tilde{H}(k)$ and residual matrix $\tilde{E}(k)$ now becomes:

$$\tilde{H}(k) = \begin{bmatrix} \frac{\partial \tilde{y}(k+1)}{\partial w_1} & \frac{\partial \tilde{y}(k+1)}{\partial w_2} & \cdots & \frac{\partial \tilde{y}(k+1)}{\partial w_{N_w}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \tilde{y}(k+H+1)}{\partial w_1} & \frac{\partial \tilde{y}(k+H+1)}{\partial w_2} & \cdots & \frac{\partial \tilde{y}(k+H+1)}{\partial w_{N_w}} \end{bmatrix}, \tag{9}$$

$$\tilde{E}(k) = \left[e(k+1) \quad e(k+2) \quad \ldots \quad e(k+H+1) \right]. \tag{10}$$

The size of matrix $\tilde{R}$ is $(L_w \times H) \times (L_w \times H)$, the size of matrix $\tilde{H}(k)$ is $(L_w \times H) \times N_w$, and the size of matrix $\tilde{E}(k)$ is $(L_w \times H) \times 1$. The remainder is identical to regular EKF. In (9)-(10) we assume $L_w = 1$, if $L_w > 1$ the method works for vector-valued time series of dimensionality L_w. However, the proposed method requires at least H more calculations at each time step in comparison classic one. Experimental research to establish it's computational cost is needed.

3 Experiments

3.1 Mackey-Glass Chaotic Process

The Mackey-Glass chaotic process is a famous benchmark for time series predictions. The discrete-time equation is given by the following difference equation (with delays):

$$x_{t+1} = (1-b)x_t + a \frac{x_{t-\tau}}{1+(x_{t-\tau})^{10}}, t = \tau, \tau+1, \ldots , \tag{11}$$

where $\tau \geq 1$ is an integer. We used the following parameters: $a = 0.1$, $b = 0.2$, $\tau = 17$ as in [4]. 500 values were used for training; the next 100 values were used for testing.

First, we trained 100 DMLP networks with one hidden layer and hyperbolic tangent activation functions using traditional EKF and BP derivatives. The training parameters for EKF were set as $\eta = 10^{-3}$ and $\mu = 10^{-8}$. The number of neurons in the hidden layer was varied from 3 to 8, the order of input tapped delay line was set $N = 5$, and the initial weights were set to small random values. Each network was trained for 50 epochs. After each epoch, MS prediction on horizon $H = 14$ on training data was performed to select the best network. This network was then evaluated on the test sequence to achieve the final MS quality result. Second, we trained 100 DMLP networks with the same initial weights using the proposed Batch EKF technique together with FPTT derivatives and evaluated their MS prediction accuracy. Third, we trained 100 NARX networks (orders of tapped delay lines: $N = 5$, $L = 5$) using EKF and BPTT derivatives to make comparisons. The results of these experiments are presented in Table 1. Normalized Mean Square Error (NMSE) was used for the quality estimations.

Table 1. Mackey-Glass problem: mean NMSE errors for different prediction horizon values H

	H=1	H=2	H=6	H=8	H=10	H=12	H=14
DMLP EKF BP	0.0006	0.0014	0.013	0.022	0.033	0.044	0.052
DMLP BEKF FPTT	0.0017	0.0022	0.012	0.018	0.022	0.027	0.030
NARX EKF	0.0010	0.0014	0.012	0.018	0.023	0.028	0.032

3.2 Santa-Fe Laser Data Series

In order to explore the capability of the global behavior of DMLP using the proposed training method, we tested it on the laser data from the Santa Fe competition. The dataset consisted of laser intensity collected from the real experiment. Data was divided to training (1000 values) and testing (100 values) subsequences. This time the goal for training was to perform long-term (H=100) MS prediction. The order of the time delay line was set to $N = 25$ as in [2], the rest was the same as in the previous experiment. The obtained average NMSE for 100 DMLP networks was 0.175 for DMLP EKF BP (classic method) versus 0.082 for DMLP BEKF FPTT (proposed method). NARX networks shows NMSE 0.131 in average.

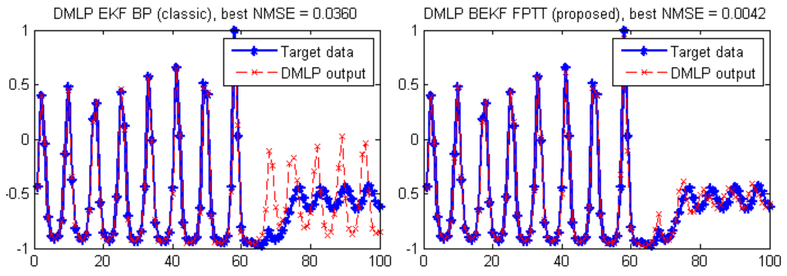


Fig. 3. The best results of the closed-loop long-term predictions (H=100) on testing data using DMLPs trained using different methods

Meanwhile, the best instance trained using Batch EKF+FPTT shows 10 times better accuracy than the best instance trained using the traditional approach.

4 Conclusions

We considered the multi-step-ahead prediction problem and discussed neural network based approaches as a tool for its solution. Feed-forward and recurrent neural models were considered, and advantages and disadvantages of their usage were discussed. A novel direct method for training feed-forward neural networks to perform multi-step-ahead predictions was proposed, based on the Batch Extended Kalman Filter. This method is considered to be useful from a technological point of view because it uses existing multi-step-ahead prediction functionality for calculating special FPTT dynamic derivatives which require a slight modification of the standard EKF algorithm. Our method demonstrates doubled long-term accuracy in comparison to standard training of the dynamic MLPs using the EKF due to direct minimization of the accumulated multi-step-ahead error.

References

1. Haykin, S.: Neural Networks and Learning Machines, 3rd edn., p. 936. Prentice Hall, New York (2009)
2. Giles, L.L., Horne, B.G., Sun-Yan, Y.: A delay damage model selection algorithm for NARX neural networks. IEEE Transactions on Signal Processing 45(11), 2719–2730 (1997)
3. Parlos, A.G., Raisa, O.T., Atiya, A.F.: Multi-step-ahead prediction using dynamic recurrent neural networks. Neural Networks 13(7), 765–786 (2000)
4. Bone, R., Cardot, H.: Advanced Methods for Time Series Prediction Using Recurrent Neural Networks. In: Recurrent Neural Networks for Temporal Data Processing, ch. 2, pp. 15–36. Intech, Croatia (2011)
5. Qina, S.J., Badgwellb, T.A.: A survey of industrial model predictive control technology. Control Engineering Practice 11(7), 733–764 (2003)
6. Toth, E., Brath, A.: Multistep ahead streamflow forecasting: Role of calibration data in conceptual and neural network modeling. Water Resources Research 43(11) (2007), doi:10.1029/2006WR005383
7. Prokhorov, D.V.: Toyota Prius HEV Neurocontrol and Diagnostics. In: Neural Networks, vol. (21), pp. 458–465 (2008)
8. Jaeger, H.: The “echo state” approach to analysing and training recurrent neural networks. Technical ReportGMDReport 148, German National Research Center for Information Technology (2001)
9. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: A Field Guide to Dynamical Recurrent Neural Networks, p. 421. IEEE Press (2001)
10. Haykin, S.: Kalman Filtering and Neural Networks, p. 304. John Wiley & Sons (2001)
11. Li, S.: Comparative Analysis of Backpropagation and Extended Kalman Filter in Pattern and Batch Forms for Training Neural Networks. In: Proceedings on International Joint Conference on Neural Networks(IJCNN 2001), Washington, DC, July 15-19, vol. 1, pp. 144–149 (2001)

Learning with Hard Constraints

Giorgio Gnecco[1], Marco Gori[2], Stefano Melacci[2], and Marcello Sanguineti[1]

[1] DIBRIS
University of Genoa, Genova, Italy
{giorgio.gnecco,marcello.sanguineti}@unige.it
[2] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche
University of Siena, Siena, Italy
{marco,mela}@dii.unisi.it

Abstract. A learning paradigm is proposed, in which one has both classical supervised examples and constraints that cannot be violated, called here "hard constraints", such as those enforcing the probabilistic normalization of a density function or imposing coherent decisions of the classifiers acting on different views of the same pattern. In contrast, supervised examples can be violated at the cost of some penalization (quantified by the choice of a suitable loss function) and so play the roles of "soft constraints". Constrained variational calculus is exploited to derive a representation theorem which provides a description of the "optimal body of the agent", i.e. the functional structure of the solution to the proposed learning problem. It is shown that the solution can be represented in terms of a set of "support constraints", thus extending the well-known notion of "support vectors".

Keywords: Learning from constraints, learning with prior knowledge, multi-task learning, support constraints, constrained variational calculus.

1 Introduction

Examples of constraints in machine learning come out naturally regardless of the context: for instance, constraints may represent prior knowledge provided by an expert (e.g., a physician in the case of a medical application: in such a case constraints may be expressed in the form of rules which help in the detection of a disease [8]). The expressive power of constraints becomes particularly significant when dealing with a specific problem, like vision, control, text classification, ranking in hyper-textual environment, and prediction of the stock market.

Table 1 provides some examples of constraints that are often encountered in practical problems arising in different domains. The first example (i) describes the simplest case in which we handle several pairs (x_i, y_i) provided for supervised learning in classification, where $y_i \in \{-1, 1\}$. If $f(\cdot)$ is the function that the agent is expected to compute, the corresponding real-valued representation of the constraint, which is reported in column 3, is just the translation of the classic "robust" sign agreement between the target and the function to be learned.

© Springer-Verlag Berlin Heidelberg 2013

Table 1. Examples of constraints from different environments

	linguistic description	*real-valued representation*
i.	i-th supervised pair for classification	$y_i \cdot f(x_i) - 1 \geq 0$
ii.	normalization of a density function	$\int_{\mathcal{X}} f(x)dx = 1$, and $\forall x \in \mathcal{X}: \ f(x) \geq 0$
iii.	coherence constraint (two classes)	$\forall x \in \mathcal{X}: \ f_1(S_1 x) \cdot f_2(S_2 x) > 0$
iv.	brightness invariance - optical flow	$\frac{\partial E}{\partial x} u + \frac{\partial E}{\partial y} v + \frac{\partial E}{\partial t} = 0$

Example *ii* is the probabilistic normalization of a density function, while example *iii* imposes the coherence between the decisions (in a binary classification problem) taken on $S_1 x$ and $S_2 x$, for the object x, where S_1 and S_2 are matrices used to select two different views of the same object x (see [9]). In the example *iv* we report a constraint from computer vision coming from the classic problem of determining the optical flow. It consists of finding the smoothest solution for the velocity field under the constraint that the brightness of any point in the movement pattern is constant. If $u(x, y, t)$ and $v(x, y, t)$ denote the components of the velocity field and $E(x, y, t)$ denotes the brightness of any pixel (x, y) at time t, then the velocity field satisfies the linear constraint indicated in Table 1 [6].

Unlike the classic framework of learning from examples, the beauty and the elegance of simplicity behind the parsimony principle - for which simple explanations are preferred to complex ones - has not been profitably used yet for the formulation of systematic theories of learning in a constrained-based environment. In those cases, most solutions are essentially based on hybrid systems, in which there is a mere combination of different modules that are separately charged of handling the prior knowledge on the tasks and of providing the inductive behavior naturally required in some tasks. In the paper, instead, we propose the study of parsimonious agents interacting simultaneously with examples and constraints in a multi-task environment with the purpose of developing the simplest (smoothest) vectorial function in a set of feasible solutions.

More precisely, we think of an intelligent agent acting on a subset $\mathcal{X}$ of the perceptual space $\mathbb{R}^d$ as one implementing a vectorial function $f := [f_1, \ldots, f_n]' \in \mathcal{F}$, where $\mathcal{F}$ is a space of functions from $\mathcal{X}$ to $\mathbb{R}^n$. Each function f_j is referred to as a *task of the agent.* As it is usual in supervised learning, we are also given a supervised learning set $\{(x_\kappa, y_\kappa),\ x_\kappa \in \mathbb{R}^d,\ y_\kappa \in \mathbb{R}^n,\ \kappa \in \mathbb{N}_{m_d}\}$.

In addition, we assume that additional prior knowledge is available, modeled by the exact fulfillment of constraints that are expressed either as

$$\forall x \in \mathcal{X}_i \subseteq \mathcal{X}: \ \phi_i(x, f(x)) = 0, \ i = 1, \ldots, m, \tag{1}$$

or as

$$\forall x \in \mathcal{X}_i \subseteq \mathcal{X}: \quad \check{\phi}_i(x, f(x)) \geq 0, \quad i = 1, \ldots, m, \tag{2}$$

where the sets $\mathcal{X}_i$ are open and $\phi_i, \check{\phi}_i$ are scalar-valued functions. We denote by $\mathcal{C}$ the collection of constraints (1) or (2). Following the terminology in variational calculus, we call (1) *hard bilateral holonomic constraints* and (2) *hard unilateral* *holonomic constraints*. Such constraints are called *hard* since they cannot be violated; constraints that can be violated (at the cost of some penalization) play the role of *soft* constraints (e.g., usually, the ones associated with the supervised pairs of the learning set). Examples of learning problems with holonomic constraints are given, e.g., in [2,5], where the constraints arise by a suitable functional representation of prior knowledge expressed in terms of first-order-logic clauses. In the paper, we investigate theoretically the problem of learning in a constrained-based environment that takes into account at the same time both the examples and constraints of holonomic type.

The paper is organized as follows. In Section 2 we formalize the problem of learning from examples with hard constraints. A representer theorem for the solution is derived in Section 3. Section 4 is devoted to the concepts of reactions of the constraints and support constraint machines. Section 5 is a short discussion.

2 Formulation of the Learning Problem

In the following, we assume $\mathcal{X}$ to be either the whole $\mathbb{R}^d$, or an open, bounded and connected subset of $\mathbb{R}^d$, with strongly local Lipschitz continuous boundary [1]. In particular, we consider the case in which, $\forall j \in \mathbb{N}_n := \{1, \ldots, n\}$ and some positive integer k, the function $f_j : \mathcal{X} \to \mathbb{R}$ belongs to the Sobolev space $\mathcal{W}^{k,2}(\mathcal{X})$, i.e., the subset of $\mathcal{L}^2(\mathcal{X})$ whose elements f_j have weak partial derivatives up to the order k with finite $\mathcal{L}^2(\mathcal{X})$-norms. So,

$$\mathcal{F} := \underbrace{\mathcal{W}^{k,2}(\mathcal{X}) \times \ldots \times \mathcal{W}^{k,2}(\mathcal{X})}_{n\,\text{times}} .$$

Finally, we assume $k > \frac{d}{2}$ since, by the Sobolev Embedding Theorem (see, e.g., [1, Chapter 4]), for $k > \frac{d}{2}$ each element of $\mathcal{W}^{k,2}(\mathcal{X})$ has a continuous representative, on which the constraints (1) and (2) can be evaluated unambiguously.

We can introduce a seminorm $\| f \|_{P,\gamma}$ on $\mathcal{F}$ by the pair (P, γ), where

$$P := [P_0, \ldots, P_{l-1}]'$$

is a suitable (vectorial) finite-order differential operator of order k with l components, and $\gamma \in \mathbb{R}^n$ is a fixed vector of positive components. Let

$$< Pf_j, Pf_j > = \| f_j \|_P^2 = \sum_{r=0}^{l-1} \int_{\mathcal{X}} (P_r f_j(x) P_r f_j(x)) dx \,,$$

$V(\cdot) := \frac{1}{2}(\cdot)^2$ denote the quadratic loss function, $\mu \geq 0$ be a fixed constant, and

$$\begin{aligned} \mathcal{L}_s(f) :=& \ \| f \|_{P,\gamma}^2 + \frac{\mu}{m_d} \sum_{l=1}^{m_d} \sum_{j=1}^{n} V(y_{i,j} - f_j(x)) \\ =& \sum_{j=1}^{n} \gamma_j < Pf_j, Pf_j > + \frac{\mu}{m_d} \sum_{l=1}^{m_d} \frac{1}{2} \sum_{j=1}^{n} (y_{i,j} - f_j(x))^2, \end{aligned} \qquad (3)$$

the objective functional to be minimized. We state the following problem.

Problem LHC (Learning from examples with Hard Constraints). *Let $\mathcal{F}_\mathcal{C} \subseteq \mathcal{F}$ be the subset of functions that belong to the given functional space $\mathcal{F}$ and are compatible with a given collection $\mathcal{C}$ of hard holonomic constraints. The problem of determining a constrained (local or global) minimizer f^o of $\mathcal{L}_s$ over $\mathcal{F}_\mathcal{C}$ is referred to as learning from the soft constraints induced by the supervised examples and the square loss, and the hard holonomic constraint collection $\mathcal{C}$.*

So, Problem LHC is a problem of learning from examples with hard holonomic constraints. Of course, generalizations of this problem can be considered, in which other combinations of the constraints and other kinds of constraints are considered [3]. If we choose for P the form used in Tikhonov's stabilizing functionals [11], for $n = 1$ and $l = k + 1$ we get

$$\| f \|_P^2 = \int_\mathcal{X} \sum_{r=0}^{k} \rho_r(x) \left(D_r f(x)\right)^2 dx \,,$$

where the function $\rho_r(x)$ is nonnegative, $P_r := \sqrt{\rho_r(x)} D_r$, and D_r denotes a differential operator with constant coefficients and containing only partial derivatives of order r. An interesting case corresponds to the choice $\rho_r(x) \equiv \rho_r \geq 0$ and

$$D_{2r} = \Delta^r = \nabla^{2r} \,, \tag{4}$$

$$D_{2r+1} = \nabla \nabla^{2r} \,, \tag{5}$$

where Δ denotes the Laplacian operator and ∇ the gradient, with the additional condition $D_0 f = f$ (see [10, 12]). According to (3), when $n > 1$ the operator P acts separately on all the components of f, i.e.,

$$Pf := [Pf_1, Pf_2, \ldots, Pf_n]' \,.$$

Note that in this case we have overloaded the notation and used the symbol P for both the vector differential operator and the scalar ones that constitute its components. We focus on the case in which the operator P is invariant under spatial shift and has constant coefficients. We use the following notation. For a function u and a multiindex α with d nonnegative components α_j, we write $D^\alpha u$ to denote $\frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}} u$, where $|\alpha| := \sum_{j=1}^{d} \alpha_j$. So, the generic component P_i of the operator P has the expression

$$P_i = \sum_{|\alpha| \leq k} b_{i,\alpha} D^\alpha \,,$$

where the $b_{i,\alpha}$'s are suitable real coefficients. Then, the formal adjoint of P is defined as the operator $P^\star = [P_0^\star, \ldots, P_{l-1}^\star]'$ whose i-th component $P_i^\star$ has the form

$$P_i^\star = \sum_{|\alpha| \leq k} (-1)^{|\alpha|} b_{i,\alpha} D^\alpha \,.$$

Finally, we define the operator $L := (P^\star)' P$ and, using again an overloaded notation, the one $\gamma L := [\gamma_1 L, \ldots, \gamma_n L]'$.

3 The Representer Theorem for Learning with Constraints

Given a set of m holonomic constraints (defined, in general, on different open subsets $\mathcal{X}_i$), we denote by $m(x)$ the number of constraints that are defined in the same point x of the domain. By $\hat{\mathcal{X}}$ we denote any open subset of $\mathcal{X}$ where the same subset of constraints is defined in all its points, in such a way that $m(x)$ is constant on the same $\hat{\mathcal{X}}$. For every set $\mathcal{X}_i$, by "$\mathrm{cl}(\mathcal{X}_i)$" we denote the closure of $\mathcal{X}_i$ in the Euclidean topology. Finally, for two vector-valued functions h_1 and h_2 of the same dimension, $h_1 \otimes h_2$ denotes the vector-valued function v whose first component is the convolution of the first components of h_1 and h_2, the second component is the convolution of the second components of h_1 and h_2, and so on, i.e., $v_i = (h_1 \otimes h_2)_i = h_{1,i} \otimes h_{2,i}$, for each index i. A constraint $\check{\phi}_i(x, f(x)) \geq 0$ is said to be *active* in x_0 at local optimality iff $\check{\phi}_i(x_0, f^o(x_0)) = 0$, otherwise it is *inactive* in x_0 at local optimality. Recall that a free-space Green's function g associated an operator O is a solution to the distributional differential equation $Og = \delta$, where δ is the Dirac distribution, centered on the origin.

The next theorem prescribes the functional representation of a local solution to Problem LHC. It is stated for a constrained local minimizer f^o. Its proof can be obtained as a variation of the one of [3, Theorem 15].

Theorem 1 (Representer Theorem for Problem LHC). *Let us consider Problem LHC in the case of $m < n$ bilateral constraints of holonomic type, which define the subset*

$$\mathcal{F}_\phi := \{f \in \mathcal{F} : \ \forall i \in \mathbb{N}_m, \ \forall x \in \mathcal{X}_i \subseteq \mathcal{X} : \ \phi_i(x, f(x)) = 0\}$$

of the functional space $\mathcal{F}$, where $\forall i \in \mathbb{N}_m : \phi_i \in \mathcal{C}^\infty(\mathrm{cl}(\mathcal{X}_i) \times \mathbb{R}^m)$. Let $f^o \in \mathcal{F}_C$ be any constrained local minimizer of the functional (3). Let us assume that for any $\hat{\mathcal{X}}$ and for every x_0 in the same $\hat{\mathcal{X}}$ we can find two permutations σ_f and σ_ϕ of the indexes of the n functions f_j and of the m constraints ϕ_i, such that the Jacobian matrix

$$\frac{\partial(\phi_{\sigma_\phi(1)}, \ldots, \phi_{\sigma_\phi(m(x_0))})}{\partial(f^o_{\sigma_f(1)}, \ldots, f^o_{\sigma_f(m(x_0))})}, \tag{6}$$

evaluated in x_0, is not singular. Suppose also that (6) is of class $\mathcal{C}^\infty(\hat{\mathcal{X}}, \mathbb{R}^n)$. Then, the following hold.

(i) There exists a set of distributions λ_i defined on $\hat{\mathcal{X}}$, $i \in \mathbb{N}_m$, such that, in addition to the above constraints, f^o satisfies on $\hat{\mathcal{X}}$ the Euler-Lagrange equations

$$\gamma L f^o(x) + \sum_{i=1}^{m} \lambda_i(x) 1_{\mathcal{X}_i}(x) \cdot \nabla_f \phi_i(x, f^o(x)) + \frac{1}{m_d} \sum_{\kappa=1}^{m_d} (f^o(x) - y_\kappa)\delta(x - x_\kappa) = 0, \tag{7}$$

where $\gamma L := [\gamma_1 L, \ldots, \gamma_n L]'$ is a spatial-invariant operator and $\nabla_f \phi_i$ is the gradient w.r.t. the second vector argument f of the function ϕ_i.

(ii) Let $\gamma^{-1}g := [\gamma_1^{-1}g, \ldots, \gamma_n^{-1}g]'$. If for all i one has $\mathcal{X}_i = \mathcal{X} = \mathbb{R}^d$ and g is a free-space Green's function of L, then f^o has the representation

$$f^o(\cdot) = \sum_{i=1}^{m} \gamma^{-1} g(\cdot) \vec{\otimes} \phi_i(\cdot, f^o(\cdot)) - \frac{1}{m_d} \sum_{\kappa=1}^{m_d} (f^o(x_\kappa) - y_\kappa)\gamma^{-1} g(\cdot - x_\kappa) , \quad (8)$$

where $g \vec{\otimes} \phi_i := g \otimes \omega_i$ and $\omega_i(\cdot) :=\uparrow \phi_i(\cdot, f^o(\cdot)) := -\lambda_i(\cdot) 1_{\mathcal{X}_i}(\cdot) \nabla_f \phi_i(\cdot, f^o(\cdot))$.

(iii) For the case of $m < n$ unilateral constraints of holonomic type, which define the subset $\mathcal{F}_{\check{\phi}} := \{f \in \mathcal{F} : \ \forall i \in \mathbb{N}_m \forall x \in \mathcal{X}_i \subseteq \mathcal{X}, \ \ \check{\phi}_i(x, f(x)) \geq 0\}$ of the functional space $\mathcal{F}$, (i) and (ii) still hold (with every occurrence of ϕ_i replaced by $\check{\phi}_i$) if one requires the nonsingularity of the Jacobian matrix (see (6)) to hold when restricting the constraints defined in x_0 to the ones that are active in x_0 at local optimality. Moreover, each Lagrange multiplier $\lambda_i(x)$ is nonpositive and equal to 0 *when the correspondent constraint is inactive in x at local optimality.*

4 Support Constraint Machines

4.1 Reactions of the Constraints

The next definition formalizes a concept that plays a basic role in the following developments.

Definition 1. *The distribution ω_i in Theorem 1 is called the* reaction of the i-th constraint *and $\omega := \sum_{i=1}^{m} \omega_i$ is the overall reaction of the given constraints.*

We emphasize the fact that the reaction of a constraint is a concept associated with the constrained local minimizer f^o. In particular, two different constrained local minimizers f^o may be associated with different reactions. A similar remark holds for the overall reaction of the constraints. Loosely speaking, under the assumptions of Theorem 1, the reaction of the i-th constraint provides the way under which such constraint contributes to the expansion of f^o. For instance, under the assumptions of Theorem 1 (ii), one has the expansion

$$f^o = \sum_{i=1}^{m} \gamma^{-1} g \ \otimes \ \omega_i = \gamma^{-1} g \ \otimes \ \omega ,$$

which emphasizes the roles of ω_i and ω in the expansion of f^o. So, solving Problem LHC is reduced to finding the reactions of the constraints.

4.2 Support Constraints

Starting from the concept of the reaction of a constraint, we now introduce the following two concepts.

Definition 2. *A* support constraint *is a constraint associated with a reaction that is different from* 0 *at least in one point of the domain $\mathcal{X}$.*

Definition 3. *A* support constraint machine *is any machine capable of finding a (local or global) solution for which the representer theorem (see Theorem 1) holds.*

So, under the assumptions of the Theorem 1, a constrained local minimizer f^o for Problem LHC can be obtained by the knowledge of the reactions associated merely with the support constraints. This motivates the use of the terminology "support constraints" as an extension of the concept of "support vectors". Interestingly, support vectors are particular cases of support constraints [3]. The connection with kernel methods arise also because, under certain conditions, the free-space Green's function g associated with the operator L is a kernel of a reproducing kernel Hilbert space (see, e.g., [4] and the references therein).

The emergence of constraints whose reaction is identically 0 at local optimality (thus, of constraints that are not support constraints) is particularly evident for the case of hard unilateral constraints. For instance, under the assumptions of Theorem 1 (iii), a constraint that is inactive at local optimality for all $x \in \mathcal{X}$ is associated with a Lagrange multiplier distribution $\lambda_i(\cdot)$ that is identically 0, so its reaction is identically 0, too. Therefore, such a constraint is *not* a support constraint.

It is interesting to discuss the case of a problem of learning from hard constraints in which one of the constraints is redundant, in the sense that the fulfillment of the other constraints guarantees its fulfillment, too. Of course, without loss of generality, such a redundant constraint can be discarded from the problem formulation and, provided that the assumptions of Theorem 1 hold, one still has the representation (8), for the constrained local minimizer f^o, where the Lagrange multiplier associated with the redundant constraint is 0 (hence, also the reaction from that constraint is 0). Therefore, we can say that the redundant constraint is *not* a support constraint.

4.3 Computational Issues

Although Problem LHC is reduced to finding the reactions of the constraints, a serious issue in the application of this recipe is that it is requires the knowledge of the Lagrange multipliers $\lambda_i(\cdot)$. Indeed, in addition to the fact that f^o has to satisfy (7), the constraints $\forall x \in \mathcal{X}, \forall i \in \mathbb{N}_m : \phi_i(x, f(x)) = 0$ must be verified, too. Such an implicit dependence makes it hard to solve the problem directly, unless special assumptions are satisfied (e.g., the linearity of the constraints, for which one obtains closed-form solutions [3]). The functional representation given by Theorem 1 (i) (see formula (8)) is a non-linear version of the classic functional equation known as the "Fredholm Equation of the II Kind". There are a number of theoretical and numerical studies on this equation, and particular attention has been devoted to the linear case [7]. In the general case, approximate reactions of the constraints can be obtained by discretizing the constraints, e.g., using unsupervised examples [3].

5 Discussion

In the paper, we have introduced a general theoretical framework of learning that involves agents acting in a constrained-based environment, for constraints of holonomic type. Our study focus on the open issue of designing intelligent agents with effective learning capabilities in complex environments where sensorial data are combined with knowledge-based descriptions of the tasks. Extensions of this work to other kinds of constraints, a general theory of learning from constraints, and specific examples are discussed in [3].

Acknowledgement. This research was partially supported by the research grant PRIN2009 "Learning Techniques in Relational Domains and Their Applications" (2009LNP494) from the Italian MURST.

References

1. Adams, R.A., Fournier, J.F.: Sobolev spaces. Academic Press (2003)
2. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Bridging logic and kernel machines. Machine Learning 86, 57–88 (2012), 10.1007/s10994-011-5243-x
3. Gnecco, G., Gori, M., Melacci, S., Sanguineti, M.: Foundations of support constraint machines. Technical report, DII-UNISI (2013)
4. Gnecco, G., Gori, M., Sanguineti, M.: Learning with boundary conditions. Neural Computation 25, 1029–1106 (2013)
5. Gori, M., Melacci, S.: Constraint verification with kernel machines. IEEE Trans. Neural Netw. Learning Syst. 24(5), 825–831 (2013)
6. Horn, B.K.P., Schunck, B.G.: Determining optical flow. Artificial Intelligence 17(1-3), 185–203 (1981)
7. Kreyszig, E.: Introductory Functional Analysis with Applications. Wiley & Sons (1989)
8. Kunapuli, G., Bennett, K.P., Shabbeer, A., Maclin, R., Shavlik, J.: Online knowledge-based support vector machines. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS, vol. 6322, pp. 145–161. Springer, Heidelberg (2010)
9. Melacci, S., Maggini, M., Gori, M.: Semi–supervised learning with constraints for multi–view object recognition. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009, Part II. LNCS, vol. 5769, pp. 653–662. Springer, Heidelberg (2009)
10. Poggio, T., Girosi, F.: A theory of networks for approximation and learning. Technical report. MIT (1989)
11. Tikhonov, A.N., Arsenin, V.Y.: Solution of ill-posed problems. W.H. Winston, Washington, D.C. (1977)
12. Yuille, A.L., Grzywacz, N.M.: A mathematical analysis of the motion coherence theory. Int. J. of Computer Vision 3, 155–175 (1989)

Bidirectional Activation-based Neural Network Learning Algorithm

Igor Farkaš and Kristína Rebrová

Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava
{farkas,rebrova}@fmph.uniba.sk

Abstract. We present a model of a bidirectional three-layer neural network with sigmoidal units, which can be trained to learn arbitrary mappings. We introduce a bidirectional activation-based learning algorithm (BAL), inspired by O'Reilly's supervised Generalized Recirculation (GeneRec) algorithm that has been designed as a biologically plausible alternative to standard error backpropagation. BAL shares several features with GeneRec, but differs from it by being completely bidirectional regarding the activation propagation and the weight updates. In pilot experiments, we test the learning properties of BAL using three artificial data sets with binary patterns of increasing complexity.

Keywords: bidirectional network, error-driven learning, activation states, binary codes association, biological plausibility.

1 Introduction

The standard error backpropagation learning algorithm [8] is known to be biologically implausible because it requires the mechanism of error propagation and it does not use locally available, activation-based variables. With this in mind, O'Reilly [4] designed Generalized Recirculation (GeneRec) algorithm that avoids the computation of error derivatives, yet can lead to error minimization. GeneRec was designed as an extension of Hinton and McClelland's model based on recirculation [2] between two layers of units (visible and hidden) with symmetric weights, which was restricted to autoassociation. To make it work, they used a four-stage activation update process. Unlike the recirculation algorithm, GeneRec is applied to a three-layer network using bidirectional interaction (only) between two layers of units (hidden and output) in a two-phase activation update process. In his work, O'Reilly experimented with several modifications of GeneRec, determined by weight update rules. For instance, he showed that the symmetry-preserving version of GeneRec (i.e. with symmetric hidden-to-output weights and symmetric weight update), combined with the so-called midpoint method, is equivalent to Contrastive Hebbian learning (CHL) for training Boltzmann machines (both in stochastic and deterministic versions) [4]. Both GeneRec and CHL are based on differences between two activation phases. Forward (minus) phase involves activation propagation from inputs toward outputs producing the network estimate of the output values. Subsequent backward (plus) phase

© Springer-Verlag Berlin Heidelberg 2013

flows in the opposite direction propagating the desired output throughout the network (see Sec. 2). We propose a bidirectional activation-based learning (BAL), which is based on the GeneRec model, but is completely symmetrical regarding the activation propagation and the weight update rules (Sec. 3). Our motivation for designing the BAL model was to implement it in our robotic mirror neuron system model, that is assumed to require the bidirectional mapping between high-level sensory and motor representations [7]. The behavior of BAL is tested in three preliminary experiments (Sec. 4).

2 GeneRec Model

The GeneRec model is a three-layer network with full connectivity between layers whose activation rules are described in Table 1, following [4]. The model has reciprocal connectivity between hidden and output layer with symmetric weights. The activation flow starts in minus phase, when the stimulus s_i is presented. Note that the net input term at the hidden layer includes the input from both visible layers before applying the sigmoid activation function $\sigma(\eta) = 1/(1 + \exp(-\eta))$. Output units produce activations o_k^- in minus phase but can also be clamped to target activations o_k^+ at the onset of plus phase. Input units can only deliver stimuli s_i at the onset of minus phase. This model was developed in the Leabra framework [3], which uses dynamic units approximating the behavior of biological neurons. O'Reilly [4] has shown that, under certain conditions, GeneRec computes the same error derivatives as Almeida-Pineda recurrent backpropagation [1,6].

Table 1. Equilibrium network variables in GeneRec model

Layer	Phase	Net Input	Activation
Input (s)	$-$	-	s_i = stimulus input
Hidden (h)	$-$	$\eta_j^- = \sum_i w_{ij} s_i + \sum_k w_{kj} o_k^-$	$h_j^- = \sigma(\eta_j^-)$
	$+$	$\eta_j^+ = \sum_i w_{ij} s_i + \sum_k w_{kj} o_k^+$	$h_j^+ = \sigma(\eta_j^+)$
Output (o)	$-$	$\eta_k^- = \sum_j w_{jk} h_j$	$o_k^- = \sigma(\eta_k^-)$
	$+$	-	o_k^+ = target output

The basic weight update rule in GeneRec is:

$$\varDelta w_{pq} = \lambda \; a_p^-(a_q^+ - a_q^-) \tag{1}$$

where a_p^- denotes the presynaptic and a_q^- denotes the postsynaptic unit activation in minus phase, a_p^+ is the presynaptic activation from plus phase (in output-to-hidden direction) and λ denotes the learning rate. The learning rule given in Eq. 1 is applied to both input-hidden and hidden-output weights. Due to the lack of space, the reader is left to consult the original paper [4] regarding the underlying math behind the derivation of the GeneRec learning rule.

3 Bidirectional Activation-based Learning

Bidirectional Activation-based Learning algorithm (BAL) shares with GeneRec the phase-based activations and unit types, but differs from it by the connectivity that allows completely bidirectional associations to be established (GeneRec focuses on input-to-output mapping). Unlike GeneRec, BAL uses two pairs of weight matrices for each activation phase. In addition, in BAL we do not use dynamical settling process but compute the activations in one step as described in Table 2.

Table 2. Activation phases and states in BAL model

Layer	Phase	Net Input	Activation
$\mathbf{x}$	F	-	x_i^{F} = stimulus
$\mathbf{h}$	F	$\eta_j^{\mathrm{F}} = \sum_i w_{ij}^{IH} x_i^F$	$h_j^{\mathrm{F}} = \sigma(\eta_j^{\mathrm{F}})$
$\mathbf{y}$	F	$\eta_k^{\mathrm{F}} = \sum_j w_{jk}^{HO} h_j^F$	$y_k^{\mathrm{F}} = \sigma(\eta_k^{\mathrm{F}})$
$\mathbf{y}$	B	-	y_k^{B} = stimulus
$\mathbf{h}$	B	$\eta_j^{\mathrm{B}} = \sum_k w_{kj}^{OH} y_k^{\mathrm{B}}$	$h_j^{\mathrm{B}} = \sigma(\eta_j^{\mathrm{B}})$
$\mathbf{x}$	B	$\eta_i^{\mathrm{B}} = \sum_j w_{ji}^{HI} h_j^{\mathrm{B}}$	$x_i^{\mathrm{B}} = \sigma(\eta_i^{\mathrm{B}})$

We avoid input-output notation of layers as used in GeneRec, because in our case not only output can be evoked by input presentation, but also vice versa. Hence, we label the two outer (visible) layers $\mathbf{x}$ and $\mathbf{y}$ and the hidden layer $\mathbf{h}$. Let forward activation be denoted by subscript F, backward activation denoted by subscript B. Then during the forward pass, the $\mathbf{x}$ units are clamped to $\mathbf{x}^{\mathrm{F}}$ and we get the activations $\mathbf{x}^{\mathrm{F}} \to \mathbf{h}^{\mathrm{F}} \to \mathbf{y}^{\mathrm{F}}$. During the backward pass, the $\mathbf{y}$ units are clamped to $\mathbf{y}^{\mathrm{B}}$ and we get the activations $\mathbf{y}^{\mathrm{B}} \to \mathbf{h}^{\mathrm{B}} \to \mathbf{x}^{\mathrm{B}}$.

The mechanism of weights update partially matches that of GeneRec. Each weight in BAL network (i.e. belonging to one of the four weight matrices) is updated using the same learning mechanism, according to which the weight difference is proportional to the product of the presynaptic (sending) unit activation a_p and the difference of postsynaptic (receiving) unit activations a_q, corresponding to two activation phases (F and B, in particular order). Namely, weights in $\mathbf{x}$-to-$\mathbf{y}$ direction (belonging to $\mathbf{h}$ and $\mathbf{y}$ units) are updated as

$$\varDelta w_{pq}^{\mathrm{F}} = \lambda \; a_p^{\mathrm{F}}(a_q^{\mathrm{B}} - a_q^{\mathrm{F}}), \tag{2}$$

where, as in the GeneRec algorithm, a_p^{F} denotes the presynaptic activity, a_q^{F} is the postsynaptic activity, and a_q^{B} denotes the postsynaptic activity from the opposite phase ($\mathbf{y}$-to-$\mathbf{h}$). Analogically, the weights in $\mathbf{y}$-to-$\mathbf{x}$ direction (belonging to $\mathbf{h}$ and $\mathbf{x}$ units) are updated as

$$\varDelta w_{pq}^{\mathrm{B}} = \lambda \; a_p^{\mathrm{B}}(a_q^{\mathrm{F}} - a_q^{\mathrm{B}}) \tag{3}$$

All units have trainable thresholds (biases) that are updated in the same way as regular weights (being fed with a constant input 1).

4 Experiments

We test the learning properties of BAL on three experiments with artificial binary data that differ in dimensionality, size of the training set and mapping complexity. We chose binary data because they simplify the assessment of network accuracy and they have properties of discrete sparse patterns (used in our intended application). For assessing the network performance, we used three quantitative measures (separately for F and B directions): (1) pattern success (patSucc), which indicates the proportion of output patterns that completely match targets, (2) bit success (bitSucc), the proportion of units matching their target, and (3) mean squared error (MSE) per neuron. Based on earlier experiments, we initialize the weights in all tests to small values from the normal distribution $\mathcal{N}(0; 1/\sqrt{n_I + 1})$, where n_I denotes the input data dimension.

4.1 4-2-4 Encoder

To compare the performance of BAL with GeneRec, we ran tests using the well-known 4-2-4 encoder task, following O'Reilly [4]. We investigated the convergence of BAL and the number of required training epochs as a function of the learning rate. Fig. 1 shows the convergence success for 100 networks and the average numbers of epochs needed. The simulations showed that convergence of BAL depends on the learning rate, with the highest number of 65% successful runs achieved for $\lambda = 0.9$. For comparison, O'Reilly [4] reports 90% success for basic GeneRec algorithm and 56% for a symmetric modification of GeneRec and its modification equivalent to CHL. In sum, probability of BAL convergence is lower than that of basic GeneRec rule, but comparable to its symmetric versions. We expect that the smaller number of successful runs is in both cases influenced by the bidirectional nature of the weight update.

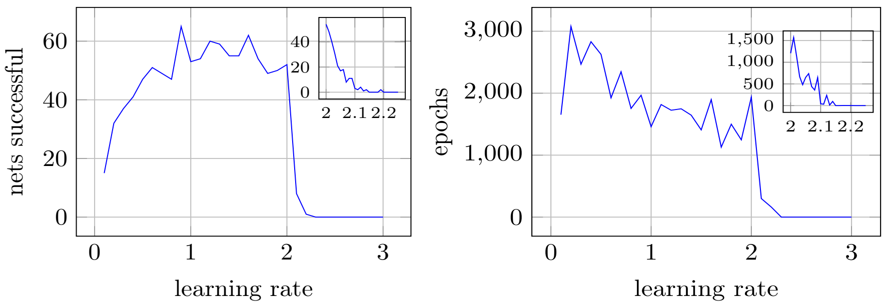


Fig. 1. 4-2-4 encoder: results for 100 nets, number of successful runs (left), average number of training epochs needed for convergence (right), both as a function of λ. Details for critical values are shown in inset plots.

BAL was observed to require a higher number of training epochs than GeneRec, with very high variability (and skewed distribution), ranging from 100

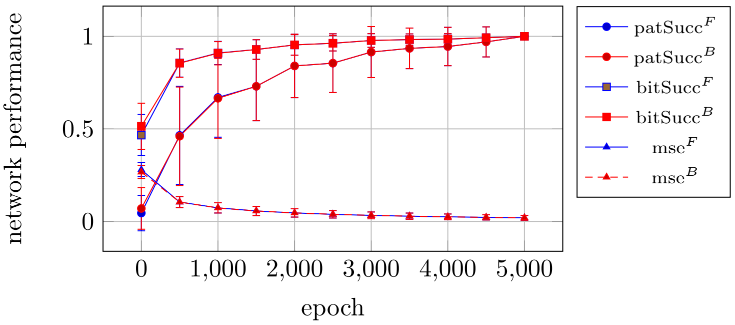


Fig. 2. Encoder 4-2-4: Development of network convergence (50 successful nets).

to thousands of epochs. On the contrary, O'Reilly reports only 418 epochs for GeneRec to converge, and less than 100 epochs for symmetric versions of GeneRec. An interesting property of BAL is that convergence probability sharply drops to zero beyond certain range of values of the learning rate, for 4-2-4 task at $\lambda = 2$. BAL convergence in 4-2-4 task and sensitivity to learning rate deserves further investigation. Fig. 2 illustrates the learning process of 50 successful networks during 5000 epochs using $\lambda = 0.9$. We conclude that MSE drops to minimum values satisfying error-free performance of the network as indicated by all success-based measures (converging to one) in both directions. If the network converges, it masters the encoder task perfectly.

4.2 Simple Binary Vector Associations

We created a sparse binary data set with high dimensionality, which resembles sensory-motor patterns from our related work. The motivation for using sparse representations comes from considering the mapping between two domains that could lend itself to generalization and robustness. In biological networks, such representations are typically achieved by lateral inhibition. As a computational shortcut, we use the k-WTA (winner-takes-all) mechanism [5]. This mechanism sets k maximally responding units to one and resets the remaining units to zero. In our related work, we apply this mechanism to output maps from two lower level modules consisting of self-organizing maps, which are then associated using the BAL algorithm. Data used in this experiment are 144-dimensional binary vectors with $k = 12$ active units. Two sets of 100 vectors are arbitrarily associated to form one-to-one mapping. Similarly to the previous experiment, we tested the network performance with various values of learning rate using 144–120–144 architecture. Fig. 3 displays the results. The network learns the mapping well up to a certain value of the learning rate ($\lambda = 0.3$), beyond which it is again observed to quickly deteriorate (Fig. 3 left). Subsequently, using the estimated optimal learning rate ($\lambda = 0.2$), we also tested selected sizes of the hidden layer n_H (Fig. 3 right). We can conclude that n_H has significant influence only on the amount of training epochs needed to reach 100% success (inset figure).

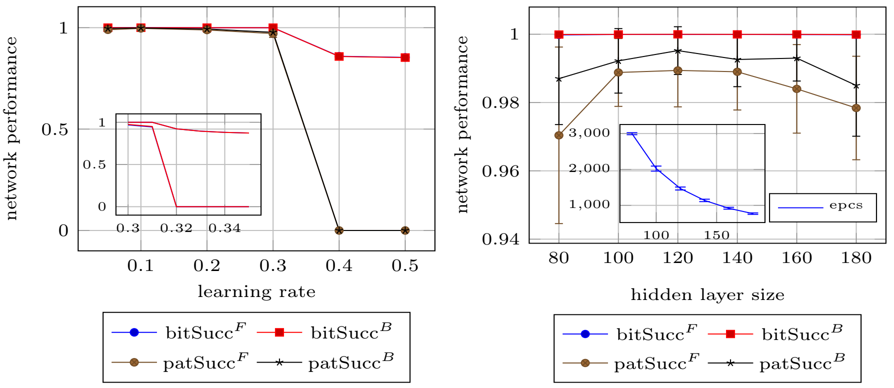


Fig. 3. Bidirectional associator: network performance as a function of λ (on the left, detail for critical values in the inset plot) and n_H (on the right, with the number of epochs needed in the inset plot).

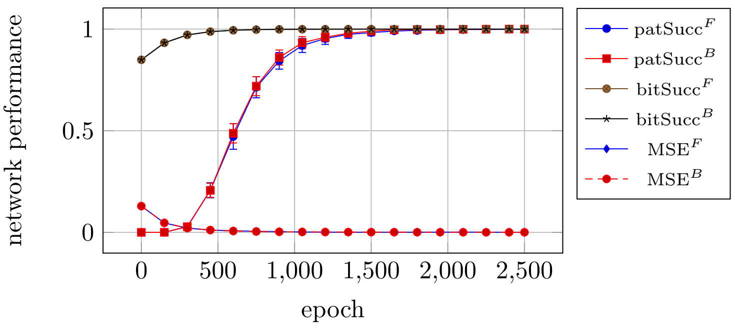


Fig. 4. Bidirectional associator: development of network performance over time (50 nets). All nets reliably converge after 1500 epochs.

To demonstrate the network training process, we computed performance measures for 50 nets trained for 2500 epochs using optimized parameters $\lambda = 0.2$ and $n_H = 120$. Results in Fig. 4 show that the networks reliably converge to successful mappings between sparse patterns. To understand the network behavior we also examined the hidden layer. We observed that $\mathbf{h}^{\mathrm{F}}$ and $\mathbf{h}^{\mathrm{B}}$ activations have a tendency to move closer to each other, as could be expected from BAL (and also from GeneRec) learning rule. Interestingly, activations of $\mathbf{h}$ units in both directions converged roughly to 0.5, so no tendency towards binary internal representations was observed. This property of internal coding is also worth further investigation.

4.3 Complex Binary Vector Associations

We evaluated the network performance on n–to–1 data associations, motivated by the sensory-motor mappings between distributed patterns. For this purpose

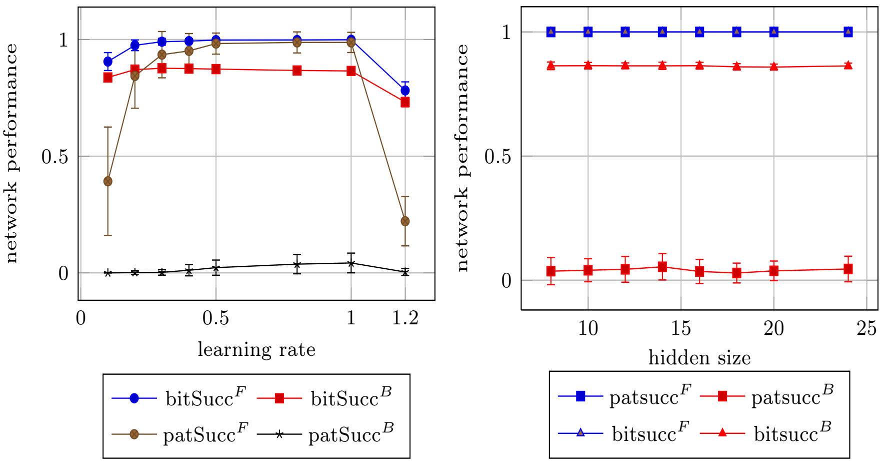


Fig. 5. Bidirectional associator with complex data: network performance as a function of λ (left) and n_H (right).

we created low-dimensional sparse binary codes, 16-dimensional vectors (4×4 map) with $k = 3$ active units with $n = 4$. For each target ($\mathbf{y}$), these four patterns ($\mathbf{x}$) were assumed to have nonzero overlap. Again, we searched for optimal λ and n_H (Fig. 5). The best performance was achieved using $\lambda \approx 1$. We can observe that the ambiguity in the data association causes the network to produce errors in B direction. For the best λ the networks yielded patSuccB ≈ 4% and bitSuccB ≈ 86%, which means that the networks made small errors in most patterns. This could be expected since the network cannot know which of the four ($\mathbf{x}$) patterns is to be reconstructed. It is known, that a network trained to associate more binary target patterns with one pattern tends to produce a mesh of outputs, weighed by their frequency of occurrence in the training set. Examples of network outputs are illustrated in Fig. 6.

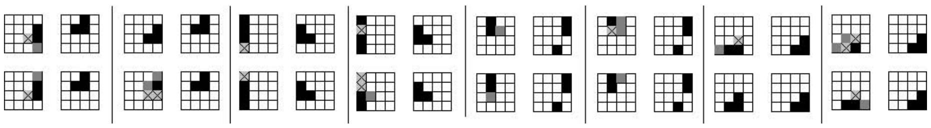


Fig. 6. Bidirectional associator with complex data: active units are filled with color, black = target–estimate match, gray = target only, gray with a cross = false-positive estimate.

5 Conclusion

We presented a new training algorithm BAL for bidirectional mappings derived from biologically plausible GeneRec model. Unlike the original GeneRec model, our model is used with standard multi-layer perceptron network without activation settling dynamics, and it learns a bidirectional mapping rather than input-output mapping. BAL also differs from the original model in the design of weight matrices, the learning rule, and partially also in the activation flow in the two activation phases. Our preliminary experiments have shown that using an appropriate learning rate, the BAL model can converge, albeit requiring more training epochs than GeneRec. In particular, for 4-2-4 encoder task the convergence is not guaranteed, which was observed also in the GeneRec model. The next step in our research should be to investigate the reasons for these performance discrepancies. Our experiments also revealed that hidden unit activations tend to converge to similar values for F and B phases. They do not tend to binarize, which is probably not necessary for learning the task. Further experiments and a more detailed analysis of BAL are required to better understand this biologically motivated bidirectional learning algorithm that will be exploited in our robotic mirror neuron system model.

Acknowledgments. This work was supported by grants 1/0439/11 from Slovak Grant Agency for Science (VEGA), and Comenius University UK/631/2013 (K.R.).

References

1. Almeida, L.: A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In: Proceedings of the IEEE First International Conference on Neural Networks, San Diego, CA (1987)
2. Hinton, G.E., McClelland, J.L.: Learning representations by recirculation. In: Neural Information Processing Systems, pp. 358–366. American Institute of Physics, New York (1988)
3. O'Reilly, R.: The Leabra model of neural interactions and learning in the neocortex. Ph.D. thesis, Carnegie Mellon University (1996)
4. O'Reilly, R.: Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. Neural Computation 8(5), 895–938 (1996)
5. O'Reilly, R., Munakata, Y., Frank, M., Hazy, T.: Computational Cognitive Neuroscience, 2nd edn. Wiki Book (2012)
6. Pineda, F.: Generalization of back-propagation to recurrent neural networks. Physics Review Letters 59, 2229–2232 (1987)
7. Rebrová, K., Pecháč, M., Farkaš, I.: Towards a robotic model of the mirror neuron system. In: The 3rd Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (2013)
8. Rumelhart, D., Hinton, G., Williams, R.: Learning internal representations by error propagation, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)

A Neural Network Model for Online Multi-Task Multi-Label Pattern Recognition

Daisuke Higuchi and Seiichi Ozawa

Graduate School of Engineering, Kobe University
1-1 Rokko-dai, Nada-ku, Kobe 657-8501, Japan

Abstract. This paper presents a new sequential multi-task learning model with the following functions: *one-pass incremental learning*, *task allocation*, *knowledge transfer*, *task consolidation*, *learning of multi-label data*, and *active learning*. This model learns multi-label data with incomplete task information incrementally. When no task information is given, class labels are allocated to appropriate tasks based on prediction errors; thus, the task allocation sometimes fails especially at the early stage. To recover from the misallocation, the proposed model has a backup mechanism called task consolidation, which can modify the task allocation not only based on prediction errors but also based on task labels in training data (if given) and a heuristics on multi-label data. The experimental results demonstrate that the proposed model has good performance in both classification and task categorization.

Keywords: multi-task learning, incremental learning, neural networks, multi-label recognition.

1 Introduction

In general, the description of an object is not unique. For example, when we look at a person's face, we might answer the parson's name. In some cases, however, we might answer gender and/or age. The object recognition with such different descriptions has often been treated as different tasks; that is, the recognition task to identify persons is called *person identification*, the recognition tasks to answer the gender and the age are called *gender recognition* and *age recognition*, respectively. Recently, the learning of multiple tasks have attracted great attentions as *multi-task learning* (MTL) [1], and it has been applied to pattern recognition called *multi-task pattern recognition* (MTPR).

Let us consider another example that a supervisor communicates with a robot to train interactively: "This person is twenty-year-old male" or "This person's age is twenty". In the first case, two class labels (*twenty-year-old* and *male*) are given by the supervisor, while no task information (*age* and *gender*) is given. Instead, in the second case, a task label (*gender*) is given with its class label (*male*). Therefore, in real situations, the class information of an object might be provided as multiple labels with/without task information. Furthermore, the tasks to be learned could be switched depending on how a supervisor gives

© Springer-Verlag Berlin Heidelberg 2013

instructions to a robot. This implies that the learning should be adaptively performed with regards to how the task and class information is given.

To deal with the abovementioned learning environments, we proposed an MTL model called *Resource Allocating Network for Multi-Task Pattern Recognition* (RAN-MTPR) [2]. In RAN-MTPR, however, it is assumed that each training data has a single class label; that is, only one recognition task is handled at a time. To alleviate this limitation, we have extended RAN-MTPR [3] such that it can learn multi-label data incrementally even when task information is not explicitly given. When no task information is given in a training data, our previous RAN-MTPR allocates the class label to the most appropriate task based on the prediction errors; thus, the task allocation sometimes fails.

In this paper, to recover from the abovementioned task misallocation, we further extend our previous RAN-MTPR so that it has a backup function called *task consolidation* which can modify the task allocation based on prediction errors, task labels in training data (if given), and the following heuristics: classes in a multi-label data belong to different tasks. In addition, the proposed RAN-MTPR has the following functions: *one-pass incremental learning*, *task allocation*, *knowledge transfer*, *learning of multi-label data*, and *active learning.* In Section 2, the main functions of the proposed RAN-MTPR are explained. Section 3 shows the experimental results on the performance evaluation using several self-defined multi-task multi-label data sets, which are originally provided as single-task problems. In Section 4, we conclude this work.

2 Extended RAN-MTPR

2.1 Learning Environments

Assume that training data of multiple tasks are given as a data stream and discarded after learning; that is, we assume *one-pass incremental learning* [4] and *sequential multi-task learning* [5]. In addition, we deal with the learning and the recognition of multi-label data under a supervised setting for class prediction. On the other hand, we assume that task information could sometimes be missing; that is, a semi-supervised setting is assumed for task allocation.

Let $(\boldsymbol{x}, \{d_l, s_l\}_{l=1}^{L})$ be a training data with L class labels, where $\boldsymbol{x}$, d_l, and s_l are an input vector, the lth class label, and its task label, respectively. If the task label s_l for class d_l is given, the training data would be learned as task s_l by using a conventional supervised learning algorithm. On the other hand, if the task information is missing, class d_l should be allocated to an appropriate task and associated with a task label s_l. The task to be allocated might be an existing task; in some cases, however, no appropriate task might be found for class d_l. In this case, the learning should be suspended until an appropriate task is found.

2.2 Network Structure

Figure 1 illustrates the network structure of RAN-MTPR, which consists of the following three components: (1) RAN classifier, (2) long-term memory (LTM),

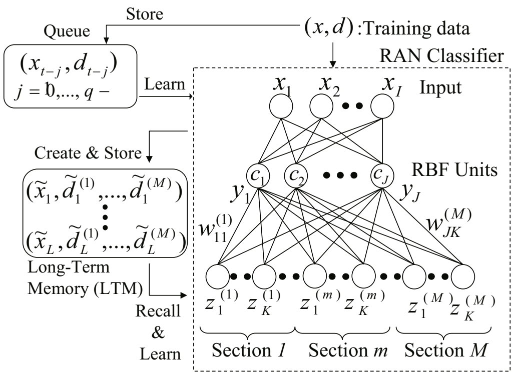

Store
Queue
(x,d) :Training data
RAN Classifier
(x_{t-j}, d_{t-j})
$j = 0,..., q-$
Learn
x_1 x_2 x_I Input
Create & Store
c_1 c_2 c_J
RBF Units
y_1 y_J
$(\tilde{x}_1, \tilde{d}_1^{(1)},..., \tilde{d}_1^{(M)})$
$(\tilde{x}_L, \tilde{d}_L^{(1)},..., \tilde{d}_L^{(M)})$
$w_{11}^{(1)}$ $w_{JK}^{(M)}$
Long-Term Memory (LTM)
Recall & Learn
$z_1^{(1)}$ $z_K^{(1)}$ $z_1^{(m)}$ $z_K^{(m)}$ $z_1^{(M)}$ $z_K^{(M)}$
Section 1 Section m Section M

Fig. 1. Network structure of RAN-MTPR

and (3) queue. The *RAN classifier* is constructed with Resource Allocating Network with Long-Term Memory (RAN-LTM) [6], which learns connection weights incrementally and adds new RBF units automatically based on Resource Allocating Network (RAN) [7]. Each output section is responsible for a specific task; thus, if RAN-MTPR learns M tasks correctly, M output sections are created (see Fig. 1). LTM is the place to store representative training data called *memory items*, which are utilized for suppressing *catastrophic forgetting* in neural network learning. When a new data is given, some memory items are retrieved from LTM and learned them together with the new data. As seen in Fig. 1, RAN-MTPR shares all RBFs among different tasks; thus, the internal representation is shared as *inductive bias* of different tasks [1,5]. The *queue* corresponds to a short-term memory to store the data whose associated tasks are unknown. If the number of data in the queue reaches its capacity q and the average error is larger than a threshold, RAN-MTPR creates a new output section to learn the queued data.

RAN-MTPR has the following functions: (1)one-pass incremental learning, (2) knowledge transfer, (3) task allocation, (4) learning of multi-label data, (5) task consolidation, and (6) active learning. Due to the space limitation, we only explain knowledge transfer and active learning. See [2,3] for the details.

2.3 One-Pass Incremental Learning

Let $\boldsymbol{x} = \{x_1, \cdots, x_I\}^T$ and $\boldsymbol{y} = \{y_1, \cdots, y_J\}^T$ be inputs and RBF outputs in RAN classifier where I and J are respectively the number of inputs and that of RBFs (see Fig. 1). And let $\boldsymbol{z}^{(m)} = \{z_1^{(m)}, \cdots, z_K^{(m)}\}^T$ $(m = 1, \cdots, M)$ be the outputs in the mth output section. Here, $z_k^{(m)}$ and K are the kth output and the number of outputs (i.e. classes) in the mth output section, respectively.

Assume that a multi-label data $(\boldsymbol{x}, \{d_l, s_l\}_{l=1}^{L})$ is given, where L $(\leq M)$ is the number of labels, d_l is the class label, and s_l is the task label. In some case, task labels s_l may not be given ($s_l = 0$ in this case). When a task label is given, the association between d_l and s_l is represented by a function: $s_l = g(d_l)$.

For the data with a task label s_l or an associated task label $s_l = g(d_l)$, RBF outputs $\boldsymbol{y}$ and the outputs $\boldsymbol{z}^{(s_l)}$ of RAN classifier are calculated by

$$y_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_j\|^2}{2\sigma^2}\right) \quad (j = 1, \cdots, J) \tag{1}$$

$$z_k^{(s_l)}(\boldsymbol{x}) = \sum_{j=1}^{J} w_{kj}^{(s_l)} y_j(\boldsymbol{x}) + \xi_k^{(s_l)} \quad (k = 1, \cdots, K) \tag{2}$$

where $\boldsymbol{c}_j = \{c_{j1}, \cdots, c_{jI}\}^T$ is the jth RBF center, σ is the RBF width, $w_{kj}^{(s_l)}$ is the connection weight from the jth RBF to the kth output in the s_lth output section, and $\xi_k^{(s_l)}$ is the bias of the kth output.

The learning is conducted by minimizing the error between $\boldsymbol{z}^{(s_l)}$ and the target vector $\boldsymbol{d}^{(s_l)} = \{d_1^{(s_l)}, \cdots, d_K^{(s_l)}\}^T$ which represents class d_l in the binary encoding. To suppress the catastrophic forgetting of RAN classifier, representative training data are stored in LTM as memory items $\Omega = (\tilde{\boldsymbol{x}}_n, \{\tilde{d}_{nl}, \tilde{s}_{nl}\}_{l=1}^{L_n})_{n=1}^{N}$, and some memory items are retrieved and learned with a training data $(\boldsymbol{x}, \{d_l, s_l\}_{l=1}^{L})$. Let $\Omega(\boldsymbol{x})$ be the index set of memory items retrieved from $\boldsymbol{x}$. Then, the total error to be minimized is shown as follow:

$$E^{(s_l)}(\boldsymbol{x}) = \|\boldsymbol{d}^{(s_l)} - \boldsymbol{z}^{(s_l)}(\boldsymbol{x})\|^2 + \sum_{n \in \Omega(\boldsymbol{x})} \|\tilde{\boldsymbol{d}}_n^{(\tilde{s}_l)} - \boldsymbol{z}^{(\tilde{s}_l)}(\tilde{\boldsymbol{x}}_n)\|^2 \tag{3}$$

where $\tilde{\boldsymbol{d}}_n^{(\tilde{s}_l)}$ and $\boldsymbol{z}^{(\tilde{s}_l)}(\tilde{\boldsymbol{x}}_n)$ are the target vector representing class d_l in the binary encoding and the output for $\tilde{\boldsymbol{x}}_n$ in the s_lth output section, respectively.

In RAN-MTPR, the connection weights $w_{kj}^{(s_l)}$ and the thresholds $\xi_k^{(s_l)}$ are learned based on the conventional gradient descent algorithm. Since all the output sections share the same RBFs, RBF centers and widths should not be modified because the learning of a certain output section may cause unexpected forgetting for other output sections.

2.4 Task Allocation and Learning of Multi-Label Data

When a multi-label data $(\boldsymbol{x}, \{d_l, s_l\}_{l=1}^{L})$ is given, it is separated into L pairs of input and class label $(\boldsymbol{x}, d_l)$ $(l = 1, \cdots, L)$, and the learning is carried out at the following two stages. At the first stage, for all pairs $(\boldsymbol{x}, d_l)$ $(l = 1, \cdots, L)$ whose task label is provided in the training data or associated with the given class label d_l (i.e. $s_l = g(d_l)$ is already defined), they are learned at the output sections corresponding to s_l.

At the second stage, for the other $(\boldsymbol{x}, d_l)$ (i.e. data without task labels), $\boldsymbol{x}$ is given to RAN classifier, and the error is calculated at the output sections which were not learned at the first stage. If there exists an output section s_l whose error is smaller than $(1-\theta)^2$, associate the class label d_l with s_l (i.e. $s_l = g(d_l)$); then, a new output unit for d_l is created and learned at the output section. Here, θ is a predefined threshold (typically 0.5). If the squared error is larger than $(1-\theta)^2$, the pair $(\boldsymbol{x}, d_l)$ is stored in the queue.

When queued data reaches its capacity q and the mean square error is larger than $1 + \theta^2$, the average squared error for the queued data is calculated at all the output sections. If there exists the output section whose average squared error is smaller than $1 + \theta^2$, the queued data are learned at the output section. Otherwise, a new output section is created and the queued data are learned.

2.5 Knowledge Consolidation

Unless task labels are given, the association between class and task labels relies on the prediction accuracy of RAN classifier. Therefore, there is no guarantee that the task allocation is always correct. In case that a class label is wrongly associated with a different task, RAN-MTPR should modify the wrong association by reallocating output units to different output sections. This process is called *task consolidation*.

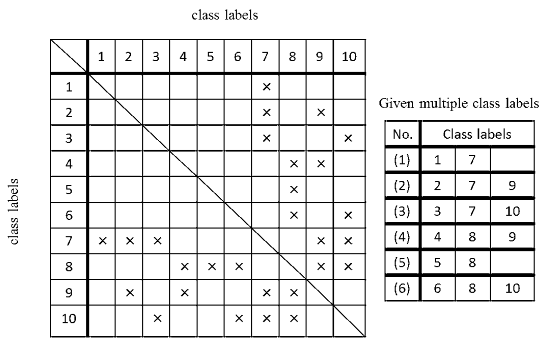
The task consolidation is evoked when more than two outputs (*conflict classes*) are activated in the same output section. Then, the consolidation process is performed as follows. For each conflict class, the prediction errors are calculated for the memory items of conflict classes at every output sections. If there is an output section where the squared error is smaller than $(1 - \theta)^2$, the output unit is reallocated to the section. Otherwise, a new output section is created and the output unit is allocated. On the other hand, if a task label is given in a training data, conflict classes should be reallocated to other sections. This reallocation is also conducted based on the prediction errors.

Finally, we introduce a new consolidation mechanism using a heuristics on multi-label data: the classes in a multi-label data belong to different tasks. When a multi-label data $(\boldsymbol{x}, \{d_l, s_l\}_{l=1}^{L})$ is given, the class labels d_l $(l = 1, \cdots, L)$ should be allocated in different output sections; that is, d_l $(l = 1, \cdots, L)$ are conflict classes. To keep the information on conflict classes, a *conflict table* is adopted. Figure 2 illustrates an example of conflict table after the six multi-label data (shown in the right-hand side) are given. For example, if a conflict class is 'class 3', the output unit for 'class 3' should not be reallocated to the output sections for 'class 7' and 'class 10'.

3 Experiments

3.1 Experimental Setup

The performance of the extended RAN-MTPR is evaluated for the five data sets in Table 1: three data sets from the UCI Machine Learning Repository [8], COIL-100 [10], and ORL face [9]. Since all the data sets are provided as single-task problems, we redefine the data sets as multi-task problems by adding different sets of class labels. Task 1 is defined by the original classification problems. As seen in Table 1, Segmentation consists of 8 classes for Task 1, and Task 2 and 3 respectively consist of 2 and 3 classes, which are redefined new tasks by combining class regions in the original task (for example, classes 1-4 in Task 1 are merged into class 8 in Task 2, and classes 5-7 in Task 1 are merged into class 9 in


class labels
class labels
Given multiple class labels

	1	2	3	4	5	6	7	8	9	10
1	\						×			
2		\					×		×	
3			\				×			×
4				\				×	×	
5					\			×		
6						\		×		×
7	×	×	×				\		×	×
8				×	×	×		\	×	×
9		×		×			×	×	\	
10			×			×	×	×		\

No.	Class labels		
(1)	1	7	
(2)	2	7	9
(3)	3	7	10
(4)	4	8	9
(5)	5	8	
(6)	6	8	10

Fig. 2. An example of conflict table

Table 1. Evaluated data sets

	#Data	#Attrib.	#Task	#Class	Task 1	Task 2	Task 3	Task 4	Task 5
Segmentation	210	19	3	12	7	2	3	-	-
Vowel Context	3772	10	3	15	11	2	2	-	-
Vehicle	188	18	3	8	4	2	2	-	-
COIL 100	200	317	5	43	20	5	7	2	9
ORL Face	400	24	3	44	40	2	2	-	-

Task 2). For COIL-100, the five tasks are defined: object name (20 classes), shape (5 classes), color (7 classes), food/non-food (2 classes), and material properties (9 classes). For ORL face, the three classification tasks are defined: person's name (40 classes), male/female (2 classes), and young/old (2 classes).

In a learning session, 720 training data are sequentially given to learn, and some data include multiple class labels and/or task labels; that is, training data always have class labels, but they might not have *task label* (the percentages of both multi-label data and task labels are fixed at 10%). To remove the performance dependency on data/task sequences, we perform 25 sessions and evaluate the average performance.

3.2 Experimental Results

In the experiments, we examine the effects of the new functions: *learning of multi-label data*, *task consolidation*, and *active learning*. For a comparative purpose, we set up two baseline models: RAN-MTPR [2] (no multi-label handling, error-base consolidation, and no active learning) and RAN-LTM (no task detection, no multi-label handling, no consolidation, no active learning). Since the two baseline models cannot handle multi-label data, only single-label data are trained (i.e. we set $p_m = 0$). Through the comparison with the first baseline model, the effect of introducing the heuristic consolidation and the active learning is discussed. However, the final classification performance can also depend on the task detection accuracy. Therefore, we decided to compare with RAN-LTM under

Table 2. Performance comparisons with the two baseline models (RAN-MTPR and RAN-LTM) in (a) classification accuracy [%] and (2) task categorization [%]

(a) Classification [%]

Model	RAN-MTPR	RAN-LTM	Extended RAN-MTPR		
p_m [%]	0	0	0	10	20
Segmentation	82.9±1.8	83.4±1.5	84.7±1.6	84.9±1.5	84.9±1.6
Vowel Context	72.7±4.4	73.9±2.1	81.4±2.1	81.8±2.2	82.2±2.2
Vehicle	74.0±1.4	74.6±1.4	77.6±1.5	77.7±1.4	77.9±1.5
COIL 100	96.6±2.1	97.0±2.0	98.6±1.8	98.6±1.7	98.7±1.7
ORL Face	65.3±12.9	79.0±1.4	88.3±1.2	88.6±1.0	88.9±1.0

(b) Task Categorization [%]

Model	RAN-MTPR	RAN-LTM	Extended RAN-MTPR		
p_m [%]	0	0	0	10	20
Segmentation	92	-	100	100	100
Vowel Context	72	-	100	100	100
Vehicle	100	-	100	100	100
COIL 100	80	-	100	100	100
ORL Face	20	-	100	100	100

the condition that task information is always given to training data; that is, this provides RAN-LTM a good advantage in task detection against the extended RAN-MTPR.

Tables 2(a)(b) show the classification accuracies and the task categorization accuracies in RAN-MTPR, RAN-LTM, and the extended RAN-MTPR. For the extended RAN-MTPR, the probabilities of multi-label data p_m are set to 0, 10 or 20 % (0% means a single-label setting). The output-margin threshold in the active learning is set to 0.25; that is, if the output margin in an output section is less than 0.25, the extended RAN-MTPR requires training data of the corresponding task to a supervisor.

As seen in Tables 2(a)(b), the extended RAN-MTPR outperforms RAN-MTPR in both average classification accuracy and average task categorization accuracy for all data sets (the difference in the average performances has statistical significance in the Welch's t test). The performance degradation in RAN-MTPR mainly originates from the low-performance task categorization (see Table 2(b)). Especially for the ORL face data, the task categorization accuracy in RAN-MPTR is only 20% because the error-based consolidation tends not to work well when the number of classes is large (40 classes in Task 1). Even in this case, the extended RAN-MTPR retains high task categorization accuracy, which leads to high classification accuracy. This indicates that the heuristic consolidation and the active learning work well. On the other hand, the extended RAN-MTPR also outperforms RAN-LTM in classification accuracy for all data sets. Since the task information is explicitly given to RAN-LTM, it is considered that this improvement comes from the introduction of the active learning and the knowledge transfer.

As seen in Table 2(a), in the extended RAN-MTPR, we cannot recognize statistical significance in classification accuracies for different percentages of multi-label data. This implies that the performance enhancement by introducing the active learning dominates over the effectiveness of learning multi-label data.

4 Conclusions

In this paper, we extend a sequential multi-task learning model called *Resource Allocating Network for Multi-Task Pattern Recognition* (RAN-MTPR) to multi-label classification problems. In the original RAN-MTPR, it is assumed that a training data includes only one class label at a time (i.e. a training data comes from only one task) under the condition that task information could be missing. Even if no task information is given in a training data, the extended RAN-MTPR can allocate a class label to the corresponding task based on the prediction errors; thus, the task allocation can fail in some cases. To recover the misallocation of tasks, the extended RAN-MTPR has a function called *task consolidation* which can modify the task allocation based on prediction errors, task labels in training data (if given), and a heuristics on multi-label data. In addition, the extended RAN-MTPR has a function of active learning which can make a query about unsure class predictions autonomously.

The experimental results demonstrate that the extended RAN-MTPR works well to enhance both the recognition accuracy and the task allocation accuracy.

References

1. Caruana, R.: Multitask Learning. Machine Learning 28, 41–75 (1997)
2. Nishikawa, H., Ozawa, S.: Radial Basis Function Network for Multitask Pattern Recognition. Neural Processing Letters 33(3), 283–299 (2011)
3. Takata, T., Ozawa, S.: A Neural Network Model for Learning Data Stream with Multiple Class Labels. In: International Conference on Machine Learning and Applications, vol. 2, pp. 35–40 (2011)
4. Kasabov, N.: Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines. Springer (2002)
5. Silver, D.L., Mercer, R.E.: The Task Rehearsal Method of Life-long Learning: Overcoming Impoverished Data. In: Cohen, R., Spencer, B. (eds.) AI 2002. LNCS (LNAI), vol. 2338, pp. 90–101. Springer, Heidelberg (2002)
6. Ozawa, S., Toh, S.L., Abe, S., Pang, S., Kasabov, N.: Incremental Learning of Feature Space and Classifier for Face Recognition. Neural Networks 6(5-6), 575–584 (2005)
7. Platt, J.: A Resource Allocating Network for Function Interpolation. Neural Computation 3, 213–225 (1991)
8. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, UC, Irvine, School of Info. and Comp. Sci. (2007)
9. `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`
10. `http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php`

Novel Feature Selection and Kernel-Based Value Approximation Method for Reinforcement Learning

Hunor Sandor Jakab and Lehel Csató

Babeş-Bolyai University, Faculty of Mathematics and Computer Science

Abstract. We present a novel sparsification and value function approximation method for on-line reinforcement learning in continuous state and action spaces. Our approach is based on the kernel least squares temporal difference learning algorithm. We derive a recursive version and enhance the algorithm with a new sparsification mechanism based on the topology maps represented by proximity graphs. The sparsification mechanism – speeding up computations – favors data-points minimizing the divergence of the target-function gradient, thereby also considering the *shape* of the target function. The performance of our sparsification and approximation method is tested on a standard benchmark RL problem.

Keywords: reinforcement learning, kernel methods, function approximation.

1 Introduction

Approximate reinforcement learning (RL) methods are algorithms dealing with real-world problems that are characterized by continuous, high dimensional state-action spaces and noisy measurements. In this article value functions for a given policy in continuous RL problems are estimated, also called *policy evaluation*. Linear regression models with non-linear features have been preferred [1,6] for approximate policy evaluation, due to their good convergence properties and relatively easy to analyze. Their main drawback is the necessity to carefully design problem-specific feature functions and we address this design issue by a nonlinear extension of the algorithm using "kernelization".[1] Kernelization in turn raises the problem of complexity, over-fitting, and increased computational cost. To avoid these problems, different sparsification mechanisms are employed. The sparsification procedure influences the accuracy and the generalization capability of the function approximator. In model free reinforcement learning, the accuracy of value functions around decision boundaries is extremely important, we thus introduce a proximity-graph based sparsification mechanism which takes into account the shape of the target function. We also present a recursive version of kernel least squares temporal difference learning (KLSTD) that uses the sparsification mechanism mentioned above.

2 Notation and Background

In RL data acquisition is defined by interacting with the environment in which we want to learn: the commands and their associated rewards defines the data base. The

[1] A survey of kernelized value function approximation methods can be found in [10]

© Springer-Verlag Berlin Heidelberg 2013

background is the Markov decision process (MDP) [7], commonly used for modeling in reinforcement learning problems. An MDP is a quadruple $M\ (S, A, P, R)$, where S is the (possibly infinite) set of states, A is the set of actions, $P(s'|s, a) : S\times S\times A \rightarrow [0, 1]$ describes the usually unknown transition probabilities, and $R(s, a) : S \times A \rightarrow \mathbb{R}$ is the instantaneous reward function defining the feedback to the learner. The decision making mechanism is modeled with an action selection *policy*: $\pi : S \times A \rightarrow [0, 1]$ is the conditional probability distribution – $\pi(a|s)$ – of taking action a in state s. The solution to the reinforcement learning problem is the *optimal policy* π^* maximizing the expected discounted cumulative reward: $\pi^* = \mathrm{argmax}_\pi E_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$. An important element of many RL algorithm is a representation of the *utility* of the state or state-action pairs as value $V_\pi(s) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s\right]$ or action-value functions $Q_\pi(s, a) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s, a_0 = a\right]$. Value functions express the expected long term discounted reward received when starting from a given state or state-action pair and following a given policy π. Value-based RL algorithms use the state-action value function $Q(\cdot, \cdot)$ to determine an optimal greedy action selection policy.

Since this paper focuses on value approximation, we consider the action selection policy to be fixed. This restriction gives rise to a so-called Markov Reward process [9] which we will refer to as MRP. Throughout the rest of this paper for ease of exposition we will omit the policy from the notation and we will use state value functions $V(\cdot)$ but the presented algorithms equally apply for state-action value functions as well.

3 Kernel Least Squares Value Approximation

As a basis of our value approximation framework we make use of the least squares temporal difference learning algorithm (LSTD) introduced in [2]. The LSTD method is based on a generalized linear approximation to the value function using the set of feature vectors $\boldsymbol{\phi}(s) = [\phi_1(s), \ldots, \phi_d(s)]^T$: $\tilde{V}(s) = \boldsymbol{\phi}(s)^T \boldsymbol{w}$, where $\boldsymbol{w} = [w_1, \ldots, w_d]^T$ is the vector of parameters and d is the feature space dimension.

The basis of temporal difference learning methods is the incremental update of the value function based on the *temporal difference error*: $\delta_{t+1} = R_{t+1} + \gamma \tilde{V}_t(s_{t+1}) - \tilde{V}_t(s_t) = R_{t+1} - (\phi(s_t) - \gamma\phi(s_{t+1}))^T \boldsymbol{w}$ and the parameter update is:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t\left(\delta_{t+1}\right)\phi(s_t).$$

In the limit of infinite updates, the algorithm leads to a parameter vector $\boldsymbol{w}$ that satisfies $E[\boldsymbol{\phi}(s_t)\delta_{t+1}] = 0$ [9]. Using a finite-size approximation with the sample average, we get the expression $\frac{1}{n}\sum_{t=0}^{n} \boldsymbol{\phi}(s_t)\delta_{t+1} = 0$ and substituting δ_{t+1} into the equation:

$$\tilde{A}\boldsymbol{w} = \tilde{b},\ \text{ where } \tilde{A} = \frac{1}{n}\sum_{t=0}^{n} \boldsymbol{\phi}(s_t)\left(\boldsymbol{\phi}(s_t) - \gamma\boldsymbol{\phi}(s_{t+1})\right)^T,\ \text{and}\quad \tilde{b} = \frac{1}{n}\sum_{t=0}^{n} \boldsymbol{\phi}(s_t)R_t.$$

When $\tilde{A}$ is invertible there is a direct solution for $\boldsymbol{w}$, as stated in [2]. The matrix $\tilde{A}$ is a sum of individual matrices the size of the feature space, and can be calculated incrementally. In [12] a kernelized version of the algorithm was presented, eliminating the need for manual feature construction. They employ the representer theorem and the kernel trick to reformulate the calculation of $\boldsymbol{w}$ with kernel functions:

$$\boldsymbol{w}^* = \tilde{A}^{-1}\tilde{b}, \text{ where } \tilde{A} = \frac{1}{n}\sum_{t=0}^{n} \boldsymbol{k}(s_t)\left[\boldsymbol{k}^T(s_t) - \gamma\boldsymbol{k}^T(s_{t+1})\right], \quad \tilde{b} = \frac{1}{n}\sum_{t=0}^{n} \boldsymbol{k}(s_t)R_t.$$

We used $k(\cdot,\cdot)$ to denote a valid kernel function and $\boldsymbol{k}(s) = \left[k(s,s_1)\ldots k(s,s_n)\right]^T$ to denote the vector of kernel values evaluated on the data-point s and the training data-points $s_i \quad i = \overline{1,n}$. The expression for the approximated value function becomes:

$$\tilde{V}(s) = \sum_{i=0}^{n} \boldsymbol{w}_i^* k(s_i, s_t) \quad s \in S \tag{1}$$

The inversion of $\tilde{A}$ is problematic: in an on-line setting inverting $\tilde{A}$ at each step is impractical; the computation time is cubic. In the next section we present a new type of sparsification mechanism to keep the matrix $\tilde{A}$ at a small fixed size. Since the sparsification depends on the *values of the inputs*, we need a *"good"* subset, the criteria presented on the next section. To speed up online learning, we use the Sherman-Woodbury formula and express the inverse of $\tilde{A}^{-1} = C$ recursively [3]:

$$C_{t+1} = C_t - C_t \frac{\boldsymbol{k}(s_t)\left[\boldsymbol{k}^T(s_t) - \gamma\boldsymbol{k}^T(s_{t+1})\right]}{1 + \left[\boldsymbol{k}^T(s_t) - \gamma\boldsymbol{k}^T(s_{t+1})\right] C_t\boldsymbol{k}(s_t)} C_t \tag{2}$$

The optimal coefficients at time-step $t+1$ can be obtained as: $w_{t+1}^* = C_{t+1}\tilde{b}$, see [6].

4 Sparsification of the Representation

Sparsification reduces the computational cost of kernel-based algorithms by finding a small set of data-points called *dictionary*, which will be used as basis set for subsequent computation. Common sparsification methods [4] use approximate linear independence (ALD) as a criterion for discarding data-points: a new point is approximated by the linear combination of dictionary points if the approximation error, defined as $\min_\alpha \| \sum_{i=1}^{d} \alpha_i \boldsymbol{\phi}(s_i) - \boldsymbol{\phi}(s^*)\|^2$, is below a predefined threshold ν (s^* is the new data point and α_i are the variational coefficients). The drawback of the ALD-based sparsification is that it ignores the target function (the function to be estimated) therefore also ignoring the data-points relevance in estimating it.

As a major contribution of this paper we present a sparsification where the spectral information contained in the Laplacian of an on-line constructed similarity graph is used when deciding on the inclusion or removal of a new data-point to the dictionary. We start by defining the unnormalized graph Laplacian as: $L = D - A$, where A is the weighted adjacency matrix of a proximity graph $G(\mathcal{E},\mathcal{V})$ with $\mathcal{E}$ and $\mathcal{V}$ denoting the set of edges and vertices respectively. $D_{i,i} = \sum_{j\neq i} A_{i,j}$ is a diagonal matrix containing the node degrees on the diagonal. Let us assume that the graph approximates the geodesic

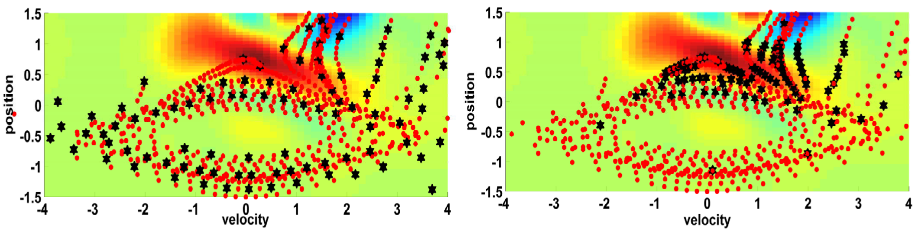


Fig. 1. Dictionary points (*black stars*) obtained after processing 5 roll-outs (≈ 450 data-points) – *red dots* – for the mountain-car control problem using different sparsification techniques. Figure (a): ALD sparsification. Figure (b) our proximity graph-based sparsification. The maximum number of dictionary points was set to 100, KLSTD approximation used.

distances[2] between data-points (details on constructing proximity graphs on-line are explained in Section 5). An interesting property of the graph Laplacian is that it is symmetric, positive semi-definite, and it gives rise to the discrete Laplacian operator Δ: for every $f : \mathcal{V} \to \mathbb{R} : \Delta f(s_i) = \sum_{j=1}^{n} A_{ij}\left[f(s_i) - f(s_j)\right]$ and $\Delta f = L\mathbf{f}$ where $\mathbf{f} = \left[f(s_1) \dots f(s_n)\right]^T$ [11].

Let us denote the approximated function f and the neighbor set of a vertex s_i as $\mathrm{neig}(s_i) = \{s_j \in \mathcal{V} | A_{i,j} \neq 0\}$. The vertices of the graph $G(\mathcal{E}, \mathcal{V})$ are samples from the target function. To obtain a sparse dictionary we introduce a score function $\mu(\cdot)$ for a vertex $s_i \in \mathcal{V}$ as the weighted sum of the squared differences between target function values of s_i and its neighboring vertices: $\mu(s_i \in \mathcal{V}) = \sum_{v_j \in \mathrm{neig}(s_i)} A_{ij}\left[f(s_i) - f(s_j)\right]^2$. Summing up the individual vertex scores gives an overall measure about the quality of the dictionary set:

$$\sum_{s_i \in \mathcal{V}} \mu(s_i) = \sum_{i,j=1}^{n} A_{ij}\left[f(s_i) - f(s_j)\right]^2 = \mathbf{f}^{\mathbf{T}}\mathbf{L}\mathbf{f} \qquad (3)$$

This means that we apply the discrete Laplacian operator for the value function, giving the divergence of the gradient of the target functions.

Whenever the size of the dictionary reaches a maximum predefined value, we compare the quality of the dictionary set with and without a new data-point using (3) and eliminate the data-point which reduces the quality the most. Figure 1 illustrates the differences between approximate linear dependence-based sparsification and our Laplacian-based method. Performing sparsification based on the maximization of the score from (3) leads to a higher sample-density in regions where f changes rapidly while at the same time keeping sufficient number of data-points in slowly changing regions to obtain good approximation accuracy.

[2] Geodesic distance is the distance between states along the manifold determined by the transition dynamics of the MRP.

5 On-Line Proximity Graph Construction

The information contained in the sequential nature of the training data in RL can be used to acquire models that reflect the transitions of the underlying MDP and the true distance between training data-points. To store this information we use so-called proximity graphs encountered in dimensionality reduction and surface reconstruction methods from point-cloud sets. Let $G(\mathcal{E}, \mathcal{V}), \quad \mathcal{V} \subset S$ be a proximity graph that is the representation of a manifold upon which the training data-points are located. Existing work [8], [11] on graph construction has focused on treating training data as either i.i.d or batch data. Since the approximation algorithm proposed by us operates on-line we present iterative versions of two well-known edge construction methods, the k nearest neighbor(kNN) and the extended sphere of influence (ϵ-SIG) methods.

The kNN edge construction strategy connects a node s_i to its k nearest neighbors $s_j \in knn(s_i)$ by an edge length equal to their spatial distance $e_{s_i,s_j} = \|s_i - s_j\|^2$. To achieve symmetry in the obtained proximity graph, we use undirected edges. As a consequence, when creating edges between data-points s_i and s_j we update the adjacency matrix as follows: $A_{i,j} = A_{j,i} = \|s_i - s_j\|^2$.

Extended Sphere of Influence Graphs $(eSIG)$*:* The extended sphere of influence graph produces a good description of non-smooth surfaces and better accommodates variations in the point sample density [8]. In this approach, edges are constructed between vertices with intersecting spheres of influence: $e_{s_i,s_j} = \|s_i - s_j\|^2$ if $R(s_i) + R(s_j) > \|s_i - s_j\|$, where $R(s_i) = \|s_i - s_k\|^2$ and s_k is the k-th nearest neighbor of s_i. The sphere of influence of a graph vertex is the sphere centered at the point, with radius given by its distance to the nearest neighbor. Disconnected components can appear with this construction, however by raising the sphere of influence radius (to be the length of the distance to the 2nd or 3rd netarest neighbor) this can be eliminated.

5.1 Updating the Graph Structure

The deletion of a vertex from the graph $G(\mathcal{E}, \mathcal{V})$ also removes the edges connecting it to its neighbors. In order to keep the information contained in the original edges about the topology we use the following pruning algorithm: Let us denote the vertex which is to be removed by s^*. After removal of s^* we get a new graph structure $G'(\mathcal{V}', \mathcal{E}')$ with the following components: $\mathcal{V}' = \mathcal{V} \setminus \{s^*\}$ and $\mathcal{E}' = \mathcal{E} \setminus \{e(s^*, s) | s \in \text{neig}(s^*)\} \cup E_{new}$ where $e(s^*, s)$ denotes an edge between the vertices s^* and s of the graph G($\mathcal{E}$,$\mathcal{V}$). The set $E_{new} = \{e(s_i, s_j) = \|s_i, s^*\| + \|s_j, s^*\| | s_i, s_j \in \text{neig}(s^*), e(s_i, s_j) \notin \mathcal{E}\}$ contains the new edges between former neighbors of s^* with weights equal to the sum of the edge weights that connected them to s^*. Figure 2 illustrates the removal procedure on a sample graph structure obtained after interacting for 400 steps with the mountain-car environment.

The addition of the new edges serves two purposes: First the connected property of the graph is maintained, secondly it can be shown that using the previously defined update the shortest path distance matrix of the graph will remain the same except for removing the row and column corresponding to the index of s^*. In consequence the geodesic distances between the remaining graph nodes are preserved.

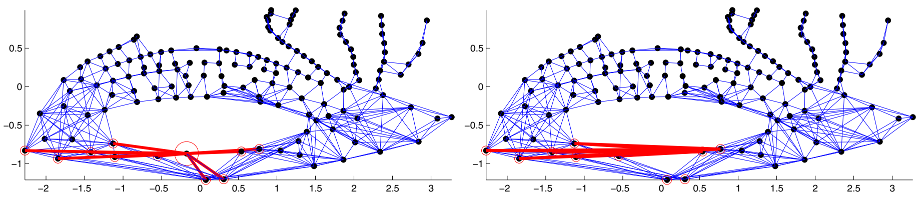
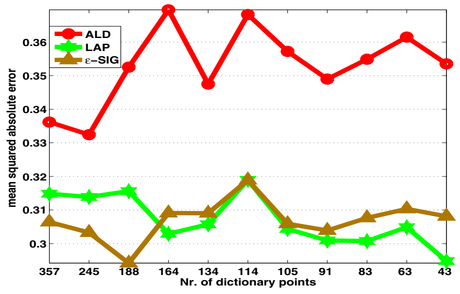


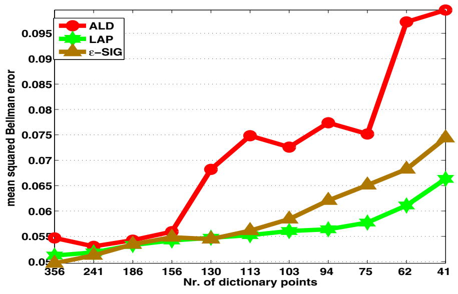
Fig. 2. Dictionary points (*black dots*) and graph structure (*blue lines*) before and after node removal. Figure (a) shows the node to be removed (*red circle*), its neighbors (*red circles with dots*) and the removable edges (*red lines*). On (b) we see the resulting graph with the newly created edges shown in red.

6 Performance Evaluation

To measure the performance of our policy evaluation framework in a continuous Markov reward process we make use of two quality measures: the Bellman Error (BE) and the mean absolute error with respect to the true value function. $BE(\tilde{V}(s)) = R(s) + \gamma \int_{s' \in S} P(s'|s,a) \int_{a \in A} \pi(a|s)\tilde{V}(s')dads' - \tilde{V}(s)$ The Bellman error expresses the ability of the approximator to predict the value of the next state based on the current state's approximated value. It also indicates how well the function approximator has converged, and according to [10] it gives an upper bound on the approximation error: $\|V - \tilde{V}\|_\infty \leq \frac{\|BE(\tilde{V})\|_\infty}{1-\gamma}$ The true value function V used as a baseline in our experiments is calculated in all experimental settings using exhaustive Monte Carlo approximation and can be considered very close to the actual value function induced by the policy π. For testing we use the well-known benchmark problems: the mountain-car [13] where the state space is two dimensional (position, and velocity), continuous, and the actions are scalars from the continuous interval [-4 , 4]. To demonstrate the benefits of our Laplacian-based sparsification mechanism for the approximation accuracy we performed 15 experiments each having 150 episodes consisting of 350 steps ($\sim$45000 training-points) on the mountain-car problem. For action selection policy π we used a Gaussian policy with a fixed linear controller: $\pi(a|s) = \frac{1}{2\pi\sigma} \exp(-\frac{\|c(s)-a\|^2}{2\sigma 2})$ where σ is a small noise term and $c(s) = \theta^T s$ is the controller with fixed parameter vector θ. We also introduced stochasticity into the dynamics of the system by adding a small Gaussian noise term with 0.02 variance to the state transitions. To see the effects of the sparsification on the approximation accuracy, we performed the same experiments for a number of different maximum dictionary sizes. Figure 3(a) shows the obtained mean absolute errors with respect to the true value function. According to our experiments our Laplacian-based sparsification mechanism leads to a lower approximation error than standard ALD for all maximum dictionary sizes. The approximation accuracy even increases slightly when the dictionary size is drastically reduced as opposed to ALD where having fewer dictionary points raises the approximation error. This may be attributed to the better placement of data-points into regions of the state-space where the target function changes more rapidly. Figure 3(b) shows the evolution of the Bellman error from the same perspective. It can be seen that the Laplacian-based sparsification

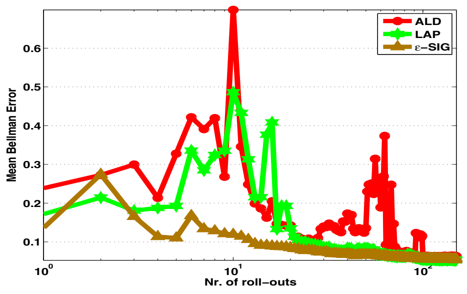


(a)

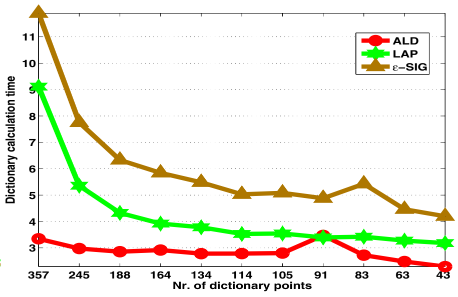


(b)

Fig. 3. Evolution of the mean absolute error for different maximum dictionary sizes averaged over 15 experiments, in case of standard *ALD* sparsification, and our Laplacian based sparsification with knn and $\epsilon - SIG$ proximity graphs. The horizontal axis represents the maximum number of dictionary points, the vertical axis on figure (a) shows the mean squared absolute error while on figure (b) the mean squared Bellman error. Measurement of the errors was done on 3600 equally distributed points from the state space.



(a)



(b)

Fig. 4. (a) Mean Bellman error as a function of training data set size. (b)Time required for the computation of the dictionary from approximately 45000 data-points using ALD and Laplacian based sparsification with knn and $\epsilon - SIG$ proximity graphs.

mechanism with knn or $\epsilon - SIG$ proximity graphs performs better at low dictionary sizes than ALD, the difference becoming more obvious as the dictionary size is further decreased. Figure 4(a) illustrates the evolution of the mean Bellman error as a function of the number of training points used for the estimation of the approximation parameters $\boldsymbol{w}$ from section 3. We calculated the dictionary beforehand based on a fixed training data set using each of the three sparsification methods. When small number of training data is used, the three dictionaries perform similarly. However when the training-data size increases the dictionary obtained by ALD leads to unstable value estimates, whereas using the dictionary obtained by our Laplacian based sparsification we obtain more accurate and stable values. The computational complexity of the presented approximation framework is influenced by the nearest neighbor search in case of graph expansion and the search for vertices with minimal score in case of the sparsification mechanism. The calculation of the individual scores of each graph vertex $v_i \in \mathcal{V}$ has a computational complexity of $O(0)$ since the scores can be stored and updated on-line for each vertex. Compared to the cost of approximate linear independence test our methods are slower as it can be seen from 4(b), but not by orders of magnitude, the difference becomes significant only by large dictionary sizes. The better approximation accuracy and faster convergence rates compensate for the higher computational requirements.

7 Conclusion

In this paper we presented an on-line algorithm with sparsification mechanism applicable to kernel-based value approximation. Our method can adjust the density of the dictionary set according to the characteristics of the target function, potentially leading to better approximation accuracy and faster convergence rates. The use of proximity graphs in the sparsification enables the extension of our methods with different distance-substitution kernels operating on graphs and opens up ways to different exploration strategies like probabilistic road-maps or rapidly exploring random trees, directions that we plan to investigate in the future.

The authors acknowledge the support of the Romanian Ministry of Education and Research via grant PN-II-RU-TE-2011-3-0278.

References

1. Boyan, J.A.: Technical update: Least-squares temporal difference learning. Machine Learning 49(2-3), 233–246 (2002)
2. Bradtke, S.J., Barto, A.G., Kaelbling, P.: Linear least-squares algorithms for temporal difference learning. In: Machine Learning, pp. 22–33 (1996)
3. Csató, L., Opper, M.: Sparse On-Line Gaussian Processes. In: Neural Computation, vol. 14(3), pp. 641–668 (2002)
4. Engel, Y., Mannor, S., Meir, R.: The kernel recursive least squares algorithm. IEEE Transactions on Signal Processing 52, 2275–2285 (2003)
5. Haasdonk, B., Bahlmann, C.: Learning with distance substitution kernels. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 220–227. Springer, Heidelberg (2004)
6. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. J. Mach. Learn. Res. 4, 1107–1149 (2003)
7. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York (1994)
8. Ruggeri, M.R., Saupe, D.: Isometry-invariant matching of point set surfaces. In: Eurographics Workshop on 3D Object Retrieval (2008)
9. Szepesvári, C.: Algorithms for Reinforcement Learning. Morgan & Claypool (2011)
10. Taylor, G., Parr, R.: Kernelized value function approximation for reinforcement learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 1017–1024. ACM, New York (2009)
11. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4) (2007)
12. Xu, X., Hu, D., Lu, X.: Kernel-based least squares policy iteration for reinforcement learning. IEEE Transactions on Neural Networks, 973–992 (2007)
13. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)

Learning of Lateral Interactions for Perceptual Grouping Employing Information Gain

Martin Meier, Robert Haschke, and Helge J. Ritter*

Neuroinformatics Group, Bielefeld University, 33501 Bielefeld, Germany
{mmeier,rhaschke,helge}@techfak.uni-bielefeld.de

Abstract. Perceptual Grouping is an important aspect in the understanding of sensory input. One of the major problems there is, how features can form meaningful groups while segregating from non relevant informations. One solution can be to couple features by attracting and repelling interactions and let neural dynamics decide the assignment of features to groups. In this paper, we present a modification of a learning approach to find these couplings, which explicitly incorporates the information gain of feature pairs, increasing the overall grouping quality of the original technique. The new approach is evaluated with an oscillator network and compared to the original work.

1 Introduction

Perceptual grouping describes the human ability to intuitively group similar features from an arbitrary domain together. These groups then have a higher level meaning. For example, in image processing, the organization of single elements in an edge filtered image to contours and shapes is an easy task from a human perspective but demanding from a computational point of view. One of the main problems that arises is how to generate "good" groups, which is closely related to the Binding Problem (see [7] for an overview).

An approach to tackle this problem is the introduction of attraction and repelling interactions between features and employ the recurrent dynamics of a neural network to yield robust grouping results, which is, for example, realized in the Competitive Layer Model (CLM) [6]. The key idea there is, that the compatibility and therefore the attraction of these features to the same group, is given by a compatibility function, which, in the easiest form, decides if the coupling is attracting or repelling in a binary fashion. This approach has been proven feasible for a broad spectrum of perceptual grouping tasks (see [9] for an overview). However, the interaction functions have to be handcrafted for each scenario and are rather complex. Also the generalization ability of these interaction functions can be poor, because the design process may be restricted to a specific detail of the problem. However, often it is simply infeasible to find an analytic description of the grouping properties.

* This work has been conducted within and funded by the German collaborative research center "SFB 673: Alignment in Communication" granted by DFG.

© Springer-Verlag Berlin Heidelberg 2013

To overcome these problems, the approach in [8] is to replace these hand-crafted interaction functions with a set of basis functions, which in turn are learned from labeled examples and simple distance measures. The basis functions in this case are estimated with a vector quantization variant, the Activity Equilibrium Vector Quantization (AEV) [1], in the space of the distance measures. The attracting respectively repelling interaction weights are obtained by incorporating the labels from the examples. Although this approach achieves good results for different perceptual grouping problems, it does not incorporate the mutual information of the elements in the proximity space, leading to a sub optimal approximation of the interaction function. To gain a better approximation, we propose to use a vector quantization variant which explicitly incorporates information theoretic principles, the ITVQ algorithm [4].

To this end, we will shortly introduce two networks for perceptual grouping tasks in the following section. First the CLM, for which the original learning algorithm was developed and later on a network composed of Kuramoto oscillators which was presented in [3] and which can achieve similar grouping results more efficient. Building on the topological properties of the CLM, the original learning algorithm is outlined in section 3 and our proposed modifications, including an abstract of the ITVQ algorithm, are stated. Following the theoretical part, we evaluated the presented techniques within a perceptual grouping scenario from [8].

2 Artificial Networks for Perceptual Grouping

In this section, we shortly introduce the architecture of two recurrent networks for perceptual grouping. At first we introduce the Competitive Layer Model, for which learning of interaction functions was originally developed. The other network utilized in the evaluation is composed of Kuramoto Oscillators [2] and was first presented in [3].

2.1 Competitive Layer Model

The CLM consists of $N \times L$ neurons which are arranged in L layers. Neurons are indexed column wise with $r = 1, \ldots, N$ denoting the feature index within each layer and $\alpha = 1, \ldots, L$ denoting the layer index. A single neuron's activity is therefore denoted as $x_{r\alpha}$. The neurons in each layer are coupled with a symmetric interaction function $f(v_r, v_{r'}) = f(v_{r'}, v_r) = f_{rr'}$ which describes the compatibility between two features v_r and $v_{r'}$. They are additionally coupled with a *winner takes all* (WTA) circuit in each column to assure that only one neuron in each column becomes active. Therefore, a single input feature v_r is represented by a column composed of L neurons $x_{r\alpha}$. Combining the lateral interaction and columnar WTA circuit, the recurrent CLM dynamics can be written as:

$$\dot{x}_{r\alpha} = -x_{r\alpha} + \sigma(J(h_r - \sum_{\beta=1}^{L} x_{r\beta}) + \sum_{r'=1}^{N} f_{rr'} x_{r'\alpha}) \; . \tag{1}$$

Here $J(h_r - \sum_\beta x_{r\beta})$ represents the WTA competition weighted by the constant J, h_r encodes the importance of feature v_r – which is set to 1 for each feature in the upcoming evaluation, because all features are equally important – and $\sigma(\cdot)$ is a linear threshold function. The lateral interaction is expressed as $\sum_{r'} f_{rr'} x_{r'\alpha}$, which calculates the support for the feature r from all other features r' in a given layer α. For a more comprehensive overview, we refer to [8,9].

2.2 Coupled Kuramoto Oscillators

The oscillator model has a topology similar to the CLM, but replaces the L neurons in each column with a single oscillator of the Kuramoto type [2], where an oscillator O_r is described by its phase θ_r and frequency ω_r. Additionally to the global coupling constant K, these oscillators are coupled individually by a symmetric matrix $M_{rr'} \equiv f(v_r, v_{r'})$. Thus, the phases θ_r of the oscillators evolve according to the following update rule:

$$\dot{\theta}_r = \omega_r + \frac{K}{N} \sum_{r'=1}^{N} f(v_r, v_{r'}) \cdot \sin(\theta_{r'} - \theta_r). \tag{2}$$

The interaction function f is limited to the interval $[-1, 1]$, where -1 and $+1$ represent strongest dissimilarity resp. similarity of features.

The oscillator frequencies are limited to discrete values $\omega_\alpha = \alpha \cdot \omega_0$, where $\alpha \in \{1, \ldots, L\}$ denotes the group index – following the CLM notation where α denotes the group/layer index. This discretization will further allow a very simple analysis of the grouping result. To cluster similar features to the same frequency ω_α, the frequency ω_r of each oscillator is updated employing the cosine similarity between the phases of the oscillators. This similarity measure is mapped to the interval $[0, 1]$, which is crucial to preserve the sign of the interaction function $f(v_r, v_{r'})$. Therefore, the frequencies are updated according to:

$$\omega_r = \omega_0 \cdot \underset{\alpha}{\operatorname{argmax}} \Big(\sum_{r' \in \mathcal{N}(\alpha)} f(v_r, v_{r'}) \cdot \frac{1}{2} \big(\cos(\theta_{r'} - \theta_r) + 1 \big) \Big), \tag{3}$$

where $\mathcal{N}(\alpha)$ denotes the set of oscillators with frequency index α, i.e. forming the current perceptual group indexed by α. This updates the frequency of an oscillator O_r to the frequency index α, whose corresponding oscillators provide most support in terms of f-weighted phase similarity. This update ensures that oscillators representing similar features will both phase-lock and converge to identical frequencies. Eq. (3) also boosts the phase-locking process, because synchronized phases do not tend to desynchronize anymore. Contrarily, oscillators representing dissimilar features will spread both in phase and frequency. The final grouping result is determined by oscillator subsets $\mathcal{N}(\alpha)$ having common frequency indices α. In terms of the CLM topology, an an oscillator O_r can represent a whole column of neurons, because it possesses two degrees of freedom. The phase θ_r represents the lateral interaction over all layers whilst the frequency ω_r acts as the layer assignment.

3 Learning Lateral Interactions

Handcrafting compatibility functions for lateral interactions can be difficult and error prone, resulting in reduced generalization abilities of these functions. To circumvent these problems, the concept in [8] is to learn these functions from labeled examples and to replace the handcrafted compatibility function by a distance function $\mathbf{d}_{rr'} = (d_1(v_r, v_{r'}), \ldots, d_n(v_r, v_{r'}))$ in a $n-$dimensional proximity space $\mathcal{D}$. This transformation assures that the required symmetry is preserved. Another important aspect of this distance function is, that it is not longer required to be one dimensional, as $f_{rr'}$ in Eq. (1). The set of distance vectors obtained from all training feature pairs is represented by a small set of representive prototypes by means of vector quantization.

The attracting respectively repelling interactions $f_{rr'}$ are created in a second step which incorporates the labels of the training data. To this end, the number of compatible and incompatible feature pairs from the training set is counted for each associated prototype. (Features are compatible if they share a common label.) These counts are finally used to determine the interaction weight c_i assigned to the prototype. For a comprehensive derivation please refer to [8], we will outline some implementation details and our proposed changes in the following sections.

3.1 Original Learning Algorithm with AEV

The vector quantization in [8] used Activity Equilibrium VQ (AEV) [1] to estimate a set of $\tilde{\mathbf{d}}_i,\ i = 1, \ldots, N$ prototypes for the distance space $\mathcal{D}$. In contrast to standard vector quantization, AEV measures the activity of each prototype based on the number of data points from the input it describes. If a prototype only represents a smaller subset of the input data than the remaining prototypes, it is repositioned in an exploration step, otherwise it is locally adapted. Combined with simulated annealing, this technique should avoid idle prototypes and place them uniformly distributed in regions of the feature space where input data is present.

After the vector quantization phase, feature pairs are sampled randomly and their labels are examined. If a feature pair shares the same label, a positive interaction c_i^+ is counted for the closest prototype $\tilde{\mathbf{d}}_i$. In the case of different labels, the interaction counts as repelling and is therefore stored as negative interaction c_i^-. After a sufficient amount of random samples, the positive and negative interactions are summed up to create the interaction coefficient

$$c_i = c_i^+ - \lambda c_i^- \tag{4}$$

for each of the $i = 1, \cdots, N$ prototypes. Here λ is a weighting parameter to account for the fact, that typically more incompatible feature pairs exist. Finally, the coefficients are normalized by the coefficient with the biggest value.

3.2 Learning Algorithm with ITVQ

Although the original learning algorithm based on AEV already yields good results, the uniform distribution of the prototypes has some drawbacks. It is desirable to find prototypes, which reflect the structure of the underlying grouping problem. To this end, we propose to replace the AEV clustering in the original algorithm with a vector quantization variant which explicitly incorporates the information density of the feature space [4]. The main idea there is to minimize the Cauchy-Schwartz (CS) divergence between the input data and the prototypes. The CS divergence measures the "distance" between two probability density functions $p(x)$ and $q(x)$ as

$$D_{CS}(p,q) = -log \frac{\int p(x)q(x)dx}{\sqrt{\int p^2(x)dx \int q^2(x)dx}} \tag{5}$$

Following the derivation in [4] by using Renyi's quadratic entropy [5] for a dataset X

$$H(X) = -log\big(V(X)\big) = -log\Big(\int p^2(x)dx\Big) \tag{6}$$

and the cross entropy between two datasets X and X_0

$$H(X, X_0) = -log\Big(\int p(x)q(x)dx\Big), \tag{7}$$

the CS divergence can be estimated by

$$D_{CS} = 2H(X, X_0) - H(X) - H(X_0). \tag{8}$$

Given the datasets X_0, which represents the original data, and the set X of prototypes, the goal is to find the dataset X which minimize the cost function

$$J(X) = \min_X D_{CS}(X, X_0). \tag{9}$$

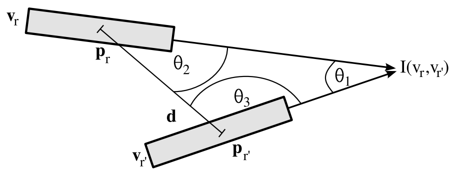
Differentiating $J(X)$ with respect to x_i and using the Parzen window technique for the dataset $X = (x_i), i = 1, \ldots, N$ with

$$p(x) = \frac{1}{N}\sum_{i=1}^{N} G_{\sigma'}(x - x_i), \tag{10}$$

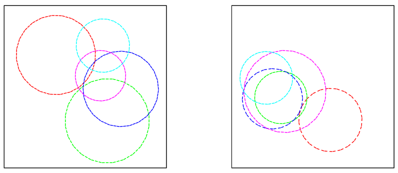
where $G_{\sigma'}(t) = e^{-\frac{t^2}{2\sigma'^2}}$ is a Gaussian kernel to estimate the pdfs of the input data, we get a simple fixed point update rule:

$$x_i^{t+1} = \frac{\sum_{j=1}^{N_0} G_\sigma(x_i^t - x_{0j})x_{0j}}{\sum_{j=1}^{N_0} G_\sigma(x_i^t - x_{0j})} - c\frac{\sum_{j=1}^{N} G_\sigma(x_i^t - x_j^t)x_j^t}{\sum_{j=1}^{N_0} G_\sigma(x_i^t - x_{0j})} + c\frac{\sum_{j=1}^{N} G_\sigma(x_i^t - x_j^t)}{\sum_{j=1}^{N_0} G_\sigma(x_i^t - x_{0j})} x_i^t \tag{11}$$

With $c = \frac{N_0}{N}\frac{V(X,X_0)}{V(X)}$. For a more comprehensive derivation please refer to [4].



(a) Oriented edge features.



(b) "Easy" problem. (c) "Hard" problem.

Fig. 1. A sketch of the parameter of the distance function is shown in 1a. Figures 1b and 1c show examples for an easy grouping task and a hard one, respectively. In the hard example, the overlapping lines in the center and on the left are nearly indistinguishable.

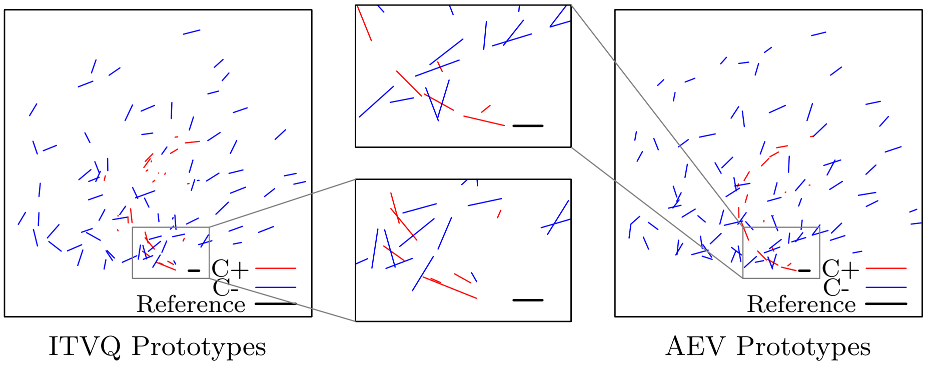


Fig. 2. Example of learned prototypes for circular training data. Red edges show attracting and blue edges represent repelling prototypes. The length of the edges encodes the interaction strength. All prototypes are positioned relative to the black feature. Although the prototypes are hard to inspect visually, ITVQ generates more positive prototypes which represent different radii of the input data. This can be seen in the zoomed part of the images. (best viewed in color)

Replacing the AEV algorithm with ITVQ for the learning of interaction prototypes while keeping the same estimation of interaction coefficients from Eq. (4) should lead to better perceptual grouping capabilities of the two networks described in section 2. We will evaluate the proposed changes with a contour grouping scenario from [8] and compare the grouping quality to the original approach in the following section.

4 Evaluation

As an evaluation scenario, we resemble the original contour grouping task from [8], where an interaction function is learned for oriented edge features. The distance between two oriented edges $d(v_r, v_{r'}) = (||p_r - p_{r'}||, \theta_1, \theta_2, \theta_3)^T$ is defined by their Euclidean distance and the three angles θ_{1-3}, as shown in Fig. 1a.

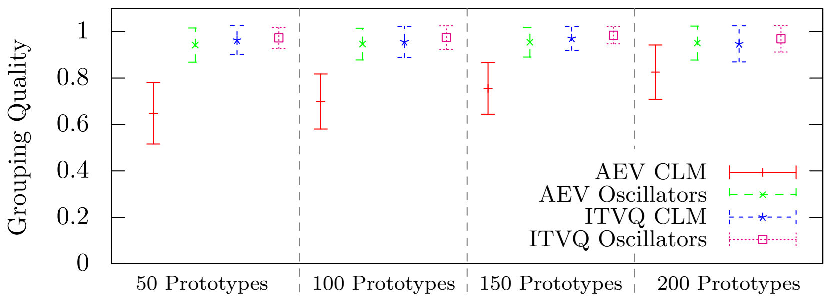


Fig. 3. This plot shows the mean grouping quality Q with standard deviation for each combination of learning algorithm and perceptual grouping network, each over 200 trials. The value is the average over all three types of shapes.

For three types of shapes, namely triangles, squares and circles, interaction functions are learned with AEV and ITVQ clustering for different numbers of prototypes. According to the findings from [8], the λ parameter from Eq. (4) is set to 2 in all trials.

In Fig. 2, 100 prototypes learned with ITVQ and AEV are shown for circular shapes composed of oriented edge features (as depcited in Fig. 1). Although the prototypes can only be evaluated qualitatively by visual inspection, the positive interaction prototypes in the zoomed areas suggest, that the ITVQ variant is more sensitive to different radii in the input data by generating more prototypes for these cases.

For the quantitative part of the evaluation, we varied the number of prototypes in steps of 50 starting with 50 for up to 200 prototypes. After learning an interaction function from eight shapes with different orientations, positions and sizes, each function was used in 200 trials to group randomly generated inputs, each input consisting of five shapes with varying sizes, positions and orientations. Because the oscillator network is less computationally demanding, we employ the same measure for counting update steps as in [3]: A single step is the update of each neuron in each layer for the CLM or the update of each oscillator for the oscillator network, respectively. Based on the previous findings, each trial was limited to a total of 500 steps. After each trial, we calculated the grouping quality Q as

$$Q = \frac{1}{N^2} \sum_{r}^{N} \sum_{r'}^{N} q_{rr'}, \quad q_{rr'} = \begin{cases} 1\, if & t_r = t_{r'} \text{ and } a_r = a_{r'} \\ 1\, if & t_r \neq t_{r'} \text{ and } a_r \neq a_{r'} \\ 0\, else & \end{cases} \tag{12}$$

where t is the target label and a the assignment generated by the perceptual grouping network. This yields a value from 0 to 1, where 1 is a perfect grouping result.

The results for this evaluation are shown in Fig. 3. Especially for a small number of prototypes, the presented modification which uses ITVQ outperforms the CLM with AEV learning by 27%. This difference decreases with an increasing number of prototypes, but the grouping is still 12% better with 200 prototypes. Also, the oscillator network achieves a better grouping with both methods, which is especially significant for the case with AEV learning. The variance of the grouping results can be explained by the random generation of the shapes. Fig. 1b shows an example of an easy grouping task, while Fig. 1c is way more demanding because of largely overlapping features. Following up to the findings of [3], we also calculated the average number of update steps for the CLM and the oscillator network which are needed to achieve the maximal grouping quality in each trail. The CLM takes an average of 88.7 steps to gain the maximal grouping quality while the oscillator network reaches this goal within an average of 24.9 steps.

5 Conclusion

We presented a modification of the learning algorithm from [8], which increases the grouping quality of perceptual networks. Evaluations with two perceptual grouping networks, the Competitive Layer Model and an oscillator network, revealed that the incorporation of ITVQ in the learning algorithm increases the grouping quality of these networks in a contour grouping task. The grouping quality is increased by more than 12% for a large number of learned prototypes. This increase is even larger for less prototypes, with a maximum increase of 27%. Another interesting point is the performance of the oscillator network in a more realistic setting than in [3]. Although the findings there suggested a good performance in perceptual grouping tasks, we could now show that its grouping capabilities exceed the CLM in the present task.

References

1. Heidemann, G., Ritter, H.: Efficient vector quantization using the wta-rule with activity equalization. Neural Processing Letters 13(1), 17–30 (2001)
2. Kuramoto, Y.: Chemical oscillations, waves, and turbulence. Dover (2003)
3. Meier, M., Haschke, R., Ritter, H.: Perceptual grouping through competition in coupled oscillator networks. In: ESANN (2013)
4. Rao, S., Han, S., Principe, J.: Information theoretic vector quantization with fixed point updates. In: International Joint Conference on Neural Networks, IJCNN 2007, pp. 1020–1024 (August 2007)
5. Rényi, A.: Some fundamental questions of information theory. Selected Papers of Alfred Renyi 2(174), 526–552 (1976)
6. Ritter, H.: A spatial approach to feature linking. In: INNC (1990)
7. Treisman, A., et al.: The binding problem. Current Opinion in Neurobiology 6(2), 171–178 (1996)
8. Weng, S., Wersing, H., Steil, J., Ritter, H.: Learning lateral interactions for feature binding and sensory segmentation from prototypic basis interactions. IEEE Transactions on Neural Networks 17(4), 843–862 (2006)
9. Wersing, H., Steil, J., Ritter, H.: A competitive-layer model for feature binding and sensory segmentation. Neural Computation 13(2), 357–387 (2001)

On–Line Laplacian One–Class Support Vector Machines

Salvatore Frandina, Marco Lippi, Marco Maggini, and Stefano Melacci

Department of Information Engineering and Mathematical Sciences,
University of Siena, Italy
{frandina,lippi,maggini,mela}@diism.unisi.it

Abstract. We propose a manifold regularization algorithm designed to work in an *on–line* scenario where data arrive continuously over time and it is not feasible to completely store the data stream for training the classifier in *batch* mode. The On–line Laplacian One–Class SVM (OLapOCSVM) algorithm exploits both positively labeled and totally unlabeled examples, updating the classifier hypothesis as new data becomes available. The learning procedure is based on conjugate gradient descent in the primal formulation of the SVM. The on–line algorithm uses an efficient buffering technique to deal with the continuous incoming data. In particular, we define a buffering policy that is based on the current estimate of the support of the input data distribution. The experimental results on real–world data show that OLapOCSVM compares favorably with the corresponding batch algorithms, while making it possible to be applied in generic on–line scenarios with limited memory requirements.

Keywords: On–line learning, One–Class SVM, RKHS, Manifold Regularization, Semi–supervised learning.

1 Introduction

Nowadays, many applications, ranging from computer vision (e.g. a camera mounted on a moving robot) to sensor networks (e.g. the measurement data collected by a sensor network), have access to a huge amount of data. In particular, many scenarios are typically *on–line*, where the data arrive continuously in time and an intelligent decision system must analyze them and take its consequent actions in synchrony with the input flow. In these cases, the capabilities of the standard *batch* machine learning algorithms are limited by the available resources both in terms of time and memory. Much attention has been paid to on–line machine learning algorithms that are able to work in an incremental and never ending learning scenario [8]. As new data becomes available, the current learned hypothesis must be continuously improved by efficiently exploiting the data seen up to the current time step and the training process must be able to adapt to eventual drifts in the data distribution. Moreover, most of the real–world problems are naturally semi–supervised, since there is only a limited quantity of labeled data and an abundant quantity of unlabeled points

© Springer-Verlag Berlin Heidelberg 2013

[2]. Goldberg et al. [5] have proposed an on–line approach to Laplacian SVMs, extending the original idea of on–line SVMs [8] to the semi–supervised setting. Different strategies are proposed to control the time and memory requirements of the algorithm.

In many real-world problems, there is an intrinsic asymmetry in the data labeling process. In general, only a few examples can be clearly and easily labeled as belonging to a given class, whereas it may be costly to provide a label for all the available data or to specify all the relationships of each example and each class. For instance, in anomaly detection tasks the evident and detected anomalies can be tagged, while most of the events will not have a clear classification. Similarly, in multi–class classification both the number of classes and their relationships (eventual intersections, inclusions and exclusions) may not be explicitly defined. Hence, we consider a setting where a classifier has to be learned only from the positive examples that describe the class to be modeled. In the context of kernel machines, this problem has been faced by One–Class SVM (OCSVM), that is essentially a density estimation algorithm sometimes referred to as *novelty detection* algorithm [14]. In this scenario, OCSVM tries to estimate the class distribution only from the available positive examples [11]. Two straightforward on–line extensions of the density estimation algorithm are proposed in [8,6].

This paper proposes an approach referred to as *On–line Laplacian One–Class SVM* (OLapOCSVM), that extends the Laplacian SVM (LapSVM) algorithm [9] to the one–class setting considering an on–line framework. In details, the continuous stream of data is efficiently bufferized by means of a pruning strategy that decides if a new example has to be added to the buffer depending on the current estimate of the support of the input data distribution. Such strategy allows us to guarantee the convergence of the learning algorithm and to exploit standard convex programming techniques, instead of a stochastic adjustment of the parameters with exponentially decaying learning rates [3,5], that may be harder to tune. In particular, we focus on the conjugate gradient descent in the primal formulation of the SVM problem [9]. The proposed buffering technique is quite general and can be applied to any semi–supervised learning algorithm.

The paper is organized as follows. The next section introduces the learning algorithms and the buffer management techniques. Then, section 3 reports the experimental comparison of the proposed algorithm with the batch approach. Finally, section 4 draws the conclusions and delineates the future developments.

2 On–Line Laplacian One–Class SVM

The considered learning scenario consists in a stream of data for which supervisions are provided only for some examples belonging to a class of interest. Hence the goal of the training algorithm is to develop the class model on–line while the data is made available, taking advantage of both the supervised and the unsupervised samples. The first aspect to be taken into account is that only positive examples are provided by the supervisions. OCSVM learning has been devised to approach this task when a batch of positive examples is available.

The learning goal is defined by the minimization of the following functional with respect to the function $f : \mathcal{X} \to I\!R$ in a given RKHS $\mathcal{H}$ and the bias $b \in I\!R$,

$$E_{\mathrm{oc}}[f, b] = \lambda_r \, \|f\|_{\mathcal{H}} + \frac{1}{|\mathcal{L}^+|} \sum_{\mathbf{x}^s \in \mathcal{L}^+} \max\left(0, 1 - f(\mathbf{x}^s) - b\right) + b \ , \qquad (1)$$

where $\mathcal{L}^+$ is the set of supervised positive examples, and $\lambda_r > 0$ weights the regularization contribution. Once the function f is estimated, an input example $\mathbf{x} \in \mathcal{X}$ is assigned to the modeled class if $f(\mathbf{x}) + b \geq 1 - \xi$, where $\xi \in [0, 1]$ is a tolerance parameter[1].

In an on–line setting the data are incrementally made available in time and the classifier should be able to provide its predictions at each time step t. Hence, the learning algorithm must be designed to yield an optimal solution at each t, given the data seen up to that instant. However, since the horizon may potentially be infinite, it is unfeasible to collect all the data up to t and to train the classifier on them. In fact, this approach would require an unbounded amount of memory and increasingly longer training times for the classifier. Moreover, the memorization of all the data history may degrade the performances in those cases in which there is a drift in the data distribution. A classical approach to on–line learning is to exploit a finite size buffer that collects only part of the incoming data [8,5]. This technique requires to define an appropriate policy to manage the buffer overflow. Depending on the specific task, when a new example has to be added to the buffer different criteria can be taken into account, especially when the buffer is full and the new example should eventually replace a memorized one. In the case of replacement, a simple criterion is to remove the oldest stored example, thus implementing a sliding window on the input stream. A better solution is to consider also the significance of the examples for training the classifier [5]. More advanced methods try to estimate the impact of each buffer replacement operation on the accuracy of the estimated classifier [12].

The proposed method is based on a pruning strategy that decides if the new example is to be added to the buffer depending on the current estimate of the support of the input data distribution. Given a tolerance $\epsilon > 0$, the example at time t, $\mathbf{x}_t$, is added to the buffer $\mathcal{B}_t$ only if the condition

$$\forall (\mathbf{x}_b \in \mathcal{B}_{t-1}) : \ \|\mathbf{x}_t - \mathbf{x}_b\|_p > \epsilon \qquad (2)$$

is satisfied, otherwise $\mathcal{B}_t = \mathcal{B}_{t-1}$. The norm used to compute the data similarity can be chosen given the specific characteristics of the input space $\mathcal{X}$ (e.g. for high–dimensional spaces $p \leq 1$ may be considered instead of the Euclidean norm with $p = 2$ [1]). This solution allows us to provide a stable coverage of the support of the input data distribution with a given resolution depending on the parameter ϵ. Clearly, a better approximation of the support is obtained for smaller values of ϵ, at the cost of a larger amount of needed memory space.

[1] This is a slightly modified formulation of the original OCSVM. The offset of 1 in the hinge loss is added to enforce a larger value of f when evaluated on the training examples and then the decision function is relaxed by ξ.

The pruning technique can be combined with replacement criteria as those listed before, when a maximum buffer size is set. In the experimental settings, we employed a buffer whose insertions are managed by the pruning technique defined by the rule of eq. (2) without limitations on its maximum size. Notice that the proposed technique allows us to filter out also potential replicates of already processed data points.

Finally, the unsupervised examples from the input stream can be exploited to provide additional information on the data distribution. By following the Laplacian SVM approach [2,9] we can add a manifold regularization term to the objective function. First, given a set of unsupervised points $\mathbf{x}^u \in \mathcal{U}$, we build the Laplacian graph $L = D - W$ associated to the set $\mathcal{M} = \mathcal{L}^+ \cup \mathcal{U}$. W is the data adjacency matrix, whose entry w_{ij} measures the *similarity* of the examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{M}$, and D is the diagonal matrix with the degree of each node, i.e. $d_{ii} = \sum_{j=1}^{|\mathcal{M}|} w_{ij}$. The manifold regularization term is defined as

$$E_{\mathrm{g}}[f] = \|f\|_{\mathcal{M}}^2 = \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ i \neq j}} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j) \right)^2 . \tag{3}$$

In general, several choices of $\|f\|_{\mathcal{M}}^2$ are possible and, in the considered case, $\|f\|_{\mathcal{M}}^2 = f^T L f$, as it can be easily derived from eq. (3) [2].

Since we are considering a one–class approach, the manifold construction is modified accordingly. In particular, the graph Laplacian must contain only the connections among the positive examples and the unsupervised ones that are most similar to them. This choice is motivated by the fact that the goal of the one–class SVM algorithm is to model the support of the distribution only for the positive samples and the presence of out–of–class–distribution data may hinder the class modeling. The unsupervised examples that have no connections in the Laplacian graph will be neglected in the learning process. We avoid using the k-NN rule to build W (as frequently done in Laplacian SVMs [2]) since it would result in connecting at least k neighbors of each point $\mathbf{x}_i$ of $\mathcal{M}$, even if some of them are far away from $\mathbf{x}_i$. Moreover the k–NN rule is known to suffer from several problems in high dimensional spaces [13]. In this work, we add a connection between two examples $\mathbf{x}_j$ and $\mathbf{x}_i$ only if $\|\mathbf{x}_i - \mathbf{x}_j\|_p < \epsilon_L$, for a given tolerance ϵ_L. The connection weight w_{ij} is then computed using an exponential decay $w_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_p / (2\sigma_L))$, where σ_L is a hyper–parameter.

This choice also allows us to efficiently handle the on–line incremental building of the adjacency matrix W. As a matter of fact, in the on–line setting W is computed on the points of $\mathcal{B}_t$, and it must be updated every time that $\mathcal{B}_t$ differs from $\mathcal{B}_{t-1}$. When a new point $\mathbf{x}$ is added to the buffer, we have already computed its distance from the other points in $\mathcal{B}_t$, due to the proposed buffering strategy (2). Since we use the same distance $\|\cdot\|_p$ both to manage the buffer and to build w_{ij}, no additional distance computations are needed to update the matrix W. Using the k-NN rule would require to store information on the k nearest neighbors of each point in $\mathcal{M}$ to update W.

Considering both the criteria of eq. (1) and eq. (3), the learning objective is to minimize the following functional with respect to f and b

$$E[f,b] = E_{\mathrm{oc}}[f,b] + \lambda_m E_{\mathrm{g}}[f] \ , \tag{4}$$

where $\lambda_m \geq 0$ weights the Laplacian contribution. By applying the Representer Theorem, the solution at the time step t can be written as $f(\mathbf{x}) = \sum_{\mathbf{x}_j \in \mathcal{B}_t} \alpha_j^t K(\mathbf{x}_j, \mathbf{x})$, where $K(\cdot,\cdot)$ is the kernel of the considered RKHS [9]. The optimal parameters α_j^t are determined by applying a conjugate gradient descent technique given the convex objective function of eq. (4). Actually, the computation is needed only when a change is applied to the buffer (insertion or replacement) and the starting point for the gradient descent is chosen such that $\alpha_j^t = \alpha_j^{t-1}$ for those examples such that $\mathbf{x}_j \in \mathcal{B}_t \cap \mathcal{B}_{t-1}$. Due to this "warm start" of the optimization procedure between consecutive time instants, only a few iterations are needed to converge to the minimum of eq. (4).

3 Experimental Results

The evaluation was performed on the MNIST and NSL–KDD datasets[2] . MNIST is a widely–used digit classification benchmark composed of 70,000 points, and we considered the classification setting used in [5], based on two binary classification problems (digit 0 vs. 1 and 1 vs. 2). The evaluation follows the one–class approach, and, hence, the classifier is trained only on the positive class, that is, digit 0 for 0 vs. 1, and digit 1 for 1 vs. 2. In the experiments, the original training and test sets were used without any preprocessing, except a normalization of the gray–level value of each pixel. NSL–KDD is an anomaly detection dataset, derived from the KDD Cup 1999 benchmark by removing several problems from the original data [15]. This dataset consists of $\approx$ 95,000 examples describing network traffic at different timestamps, each one labelled with one of following classes: DoS, R2L, U2R, probing and normal (no attack). In the experiments, the R2L class was neglected, as it contains too few examples (0.07% for the training set and 0.37% for the test set). Each continuous attribute was discretized into 10 bins, by selecting the quantization thresholds using a maximum entropy principle (each bin should contain roughly the same number of examples). Both the datasets come divided into a training and a test split. The test split was used to evaluate the classifier quality, whereas we divided the training split into two portions. The first one (10% of the data) was used in an off–line setting to cross-validate the hyper–parameters of the classifiers[3]. The second portion (90% of the data) was streamed in an on–line fashion and processed by the proposed algorithm. In both cases, only 10% of the points were supervised. A Gaussian kernel was selected and its width σ_K was chosen in a discrete grid defined in $[3, 21]$ with step size 3; the regularization parameter λ_r was selected in $\{10^{-h} | h = 1, \ldots, 6\}$;

[2] `http://yann.lecun.com/exdb/mnist/`; `http://iscx.ca/NSL-KDD/`

[3] Two thirds of the validation examples were exploited to train the classifier, whereas one third of the validation examples were used to evaluate the classifier accuracy.

the Laplacian regularization parameter λ_m was chosen in $\{0, 10^{-h}|h = 0, \ldots, 6\}$; the width of the Gaussian for computing the Laplacian graph weights σ_L and the distance tolerance ϵ_L were chosen in $[3, 12]$ with a step of 3 for the MNIST dataset, and in $\{0.5, 1, 1.5, 2, 3\}$ for the NSL–KDD dataset.

The first experiment was aimed at evaluating different settings of the pruning policy used for the buffer management. In particular, three different norms were considered (i.e. $p \in \{0.5, 1, 2\}$) and the resulting size of the buffer was computed by varying the resolution ϵ used to approximate the data distribution support. Table 1 reports the total number of elements N of the streamed training set that are stored into the buffer for the different norms and tolerance values. In particular, the table shows that larger values of ϵ are needed for the L_1 and $L_{0.5}$ norms to yield the same number of stored examples with respect to the L_2 norm. This behavior confirms the property that, when the norm index p decreases, the average distances among the points tend to increase [1]. By an appropriate choice of the combination of the norm and the related tolerance ϵ, three different settings were defined for the following experiments. In particular, in *Online setting 1* the tolerance is chosen such that roughly the 15% of the streamed examples are stored in the buffer for training the classifier; in *Online setting 2* the buffer contains about the 10% of the data, whereas in *Online setting 3* only the 5% of the available examples is memorized into the buffer.

Table 1. Number of examples N stored into the buffer with respect to the norm (L_2,L_1 and $L_{0.5}$) and the tolerance parameter ϵ, for the MNIST and NSL–KDD datasets. The pairs (norm,ϵ) define three different buffer configurations exploited in the evaluation of the learning algorithm.

	MNIST 0vs.1		MNIST 1vs.2		NSL–KDD		Buffer-Size
Norm	ϵ	N	ϵ	N	ϵ	N	
L_2	5.5	1712	6.3	1735	0.075	11195	Online setting 1 15% of the examples
L_1	55	1620	64	1699	0.12	11049	
$L_{0.5}$	7700	1679	9000	1775	0.35	11087	
L_2	6	1105	6.7	1133	0.15	8000	Online setting 2 10% of the examples
L_1	60	1165	70	1127	0.25	7723	
$L_{0.5}$	9000	1089	10500	1114	0.75	7987	
L_2	6.7	533	7.3	584	1	3850	Online setting 3 5% of the examples
L_1	70	569	80	559	1.25	3920	
$L_{0.5}$	11000	574	13000	529	5	3837	

Table 2 reports the classification performance of the OLapOCSVM algorithm on the MNIST and NSL–KDD datasets using the F1 score. The first group of results, referred to as *Batch setting*, corresponds to the (offline) batch training of the classifier on all the available training data. In the case of the NSL–KDD dataset it was not possible to train the classifier due to the large memory requirements needed to manage all the data. As a matter of fact, the proposed buffering strategy is not used and the training examples are not pruned. From a

Table 2. Accuracy of the OLapOCSVM algorithm on the MNIST and NSL–KDD datasets measured by the F1 score for each norm and ϵ setting as defined in Table 1. *Batch setting* corresponds to the (offline) batch training on the all the available data.

		MNIST		NSL–KDD		
	Norm	0vs.1	1vs.2	DoS	U2R	probing
Batch setting	L_2	0.998	0.950	-	-	-
	L_1	0.943	0.992	-	-	-
	$L_{0.5}$	0.648	0.990	-	-	-
Online setting 1	L_2	0.980	0.981	0.929	0.361	0.692
	L_1	0.954	0.987	0.923	0.353	0.716
	$L_{0.5}$	0.642	0.991	0.910	0.398	0.716
Online setting 2	L_2	0.973	0.980	0.929	0.361	0.687
	L_1	0.933	0.989	0.924	0.564	0.713
	$L_{0.5}$	0.895	0.984	0.913	0.391	0.720
Online setting 3	L_2	0.975	0.977	0.933	0.361	0.669
	L_1	0.965	0.974	0.930	0.413	0.712
	$L_{0.5}$	0.716	0.965	0.918	0.353	0.718

comparison of the results obtained in the three different on–line settings, it can be noticed that the F1 score decreases when using a larger tolerance in the data support approximation, but the performances degrade only slightly whereas the memory requirements are considerably reduced. In particular, there is not a significant performance difference with respect to the batch case, showing that the applied buffer policy is able to accurately approximate the original data distribution. Our results in the MNIST data are roughly comparable to the ones of [5], even if [5] exploits both positive and negative supervisions. We also tried to replicate the algorithm described in [5] using a OCSVM, but we were not able to obtain satisfactory performances, maybe due to difficulties in choosing an appropriate value for the learning rate used in the stochastic gradient descent.

The main advantage of the proposed approach is its capability to manage large datasets with reasonable time and memory requirements without a significant decrease of the performances. It does not require an accurate tuning of the learning rate [5], even if it may be tricky to optimize the tolerance parameter ϵ to have a good compromise between performance and resource requirements.

4 Conclusions

We presented On–line Laplacian One–Class SVM, a learning algorithm that updates a one–class classifier while new training data becomes available. In particular, the proposed buffer management policy allows the use of the algorithm both in on–line scenarios and in cases where a huge quantity of data is available. The results on the MNIST and NSL–KDD datasets show encouraging performances with a significant reduction of the storage requirements. Future work will consider an extension to other on–line semi–supervised learning algorithms, such as S3VM [7], SBRS [4], and the application of the buffering strategy to online clustering by MEEs [10]. The proposed approach will be applied to computer vision tasks in a continuous learning framework.

Acknowledgements. We thank Marco Gori for the fruitful discussions in the development of the ideas presented in this paper. This research was partially supported by the research grant PRIN2009 Learning Techniques in Relational Domains and Their Ap- plications (2009LNP494) from the Italian MURST.

References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 420–434. Springer, Heidelberg (2000)
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. The Journal of Machine Learning Research 7, 2399–2434 (2006)
3. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Compstat. pp. 177–186 (2010)
4. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Bridging logic and kernel machines. Machine learning 86(1), 57–88 (2012)
5. Goldberg, A., Li, M., Zhu, X.: Online manifold regularization: A new learning setting and empirical study, pp. 393–407. Springer (2008)
6. Gretton, A., Desobry, F.: On-line one-class support vector machines. an application to signal segmentation. In: Proceedings of Acoustics, Speech, and Signal Processing, vol. 2, pp. II–709. IEEE (2003)
7. Joachims, T.: Transductive inference for text classification using support vector machines. In: Proceedings of ICML, pp. 200–209. Morgan Kaufmann (1999)
8. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. IEEE Transactions on Signal Processing 52(8), 2165–2176 (2004)
9. Melacci, S., Belkin, M.: Laplacian Support Vector Machines Trained in the Primal. Journal of Machine Learning Research 12, 1149–1184 (2011)
10. Melacci, S., Gori, M.: Unsupervised learning by minimal entropy encoding. IEEE Transactions on Neural Networks and Learning Systems 23(12), 1849–1861 (2012)
11. Muñoz-Marí, J., Bovolo, F., Gómez-Chova, L., Bruzzone, L., Camp-Valls, G.: Semisupervised one-class support vector machines for classification of remote sensing data. IEEE Trans. on Geoscience and Remote Sensing 48(8), 3188–3197 (2010)
12. Orabona, F., Castellini, C., Caputo, B., Jie, L., Sandini, G.: On-line independent support vector machines. Pattern Recognition 43(4), 1402–1412 (2010)
13. Radovanović, M., Nanopoulos, A., Ivanović, M.: Hubs in space: Popular nearest neighbors in high-dimensional data. The Journal of Machine Learning Research 9999, 2487–2531 (2010)
14. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. 13(7), 1443–1471 (2001)
15. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: Proceedings of the International Conference on Computational Intelligence for Security and Defense Applications, pp. 53–58 (2009)

OSA: One-Class Recursive SVM Algorithm with Negative Samples for Fault Detection

Mikhail Suvorov, Sergey Ivliev, Garegin Markarian,
Denis Kolev, Dmitry Zvikhachevskiy, and Plamen Angelov

School of Computing and Communications,
Lancaster University, LA14WA, United Kingdom
p.angelov@lancaster.ac.uk

Abstract. In this paper a novel one-class classification approach (called OSA) is proposed. The algorithm is particularly suitable for fault detection in complex technological systems, such as aircraft. This study is based on the capability of one-class support vector machine (SVM) method to classify correctly the observation and measurement data, obtained during the exploitation of the system such as airborne aircraft into a single class of 'normal' behavior and, respectively, leave data that is not assigned to this class as suspected anomalies. In order to ensure real time (in flight) application a recursive learning procedure of the method is proposed. The proposed method takes into account both "positive"/"normal" and "negative"/"abnormal" examples of the base class, keeping the overall model structure as an outlier-detection approach. This approach is generic for any fault detection problem (for example in areas such as process control, computer networks, analysis of data from interrogations, etc.). The advantages of the new algorithm based on OSA are verified by comparison with several classifiers, including the traditional one-class SVM. The proposed approach is tested for fault detection problem using real flight data from a large number of aircraft of different make (USA, Western European as well as Russian).

Keywords: Flight Data Analysis, Fault Detection and Identification, One-class SVM.

1 Introduction

In this paper we propose a new approach for one-class classification using support vector machines (SVM) principles [11]. One-class classification technique represents a wide set of different machine learning approaches that assume a single class solution of a (fault detection or other) problem containing majority of the samples [14].

One-class classification method is widely applied for fault identification in complex technical systems [14]. The "normal" system performance is defined as a "base" class under the assumption that the data produced by the normally functioning system have a more structured pattern as compared to a rather random pattern of anomalies. It is also important to note that the set of all

© Springer-Verlag Berlin Heidelberg 2013

possible system faults is difficult, if possible at all, to obtain. Furthermore, even if such a set is known, it is not realistic to expect it to be a structured one. In reality, the number of all possible faults is huge and the system behavior in case of a hypothetical combination of different faults is, generally, unpredictable. Therefore, it is a real problem to generate a sample training set for the algorithm that has a generalization capability for description of the faults. For that reason one class classification problems are considered as an alternative.

Often, conventional data analysis algorithms of the technical systems (for example, fault detection in flight data) are based on the set of predefined logical rules (that use thresholds and conditions). For example, in flight data analysis (FDA) systems such as express analysis, EA by SKAT [15] or Automated Ground Station, AGS by Sagem [16] each possible event/fault is listed and special logical condition is mapped to it. During the analysis process registered data is verified for the appearance of such logical conditions. If any condition is satisfied, the corresponding event is declared.

Usually, such logical conditions involve a limited subset of registered physical variables (features) and a large number of thresholds, predefined by the aircraft manufacturer (in the case of a FDA system). Such an approach has several systemic drawbacks. The most significant one is the fact, that often manual fitting of the rules thresholds is required in order to adapt the algorithm to the corresponding machine/aircraft. A limited event list does not allow the detection of any novel event, interaction between features values are not taken into account [17].

Additional requirement for the algorithm that can be essential for in-flight real time fault detection is the "on-line" operation mode. This includes recursive nature of the mathematical expressions with no need to store/memorize past data but to accumulate instead the statistically aggregated information, and the ability to update recursively the parameters of the algorithm itself [12].

One of the most popular approaches in one-class classification is support vector machine (SVM) algorithm [1]. Initially proposed as a linear classifier, it is possible to introduce non-linearity in the algorithm using the well-known "kernel trick" [11]. The algorithm provides analytically optimal solution for which an incremental learning procedure can also be developed. The standard one-class SVM method [6] is trained over the data set that contains only "base" class examples ("normal", not faulty one).

One-class classification problem can also be considered as outlier/anomaly detection problem [4,6]. This type of approaches is convenient in the cases, when most of the data can be associated with a single class. This subset of the data is usually described as the "base" class (the class of "normality"). All other data usually have too complex pattern or are not in large amount in the training sample. Typically, the one-class classifier [4,6,10] takes as an input a vector of the data samples and produces as an output the label of that data sample/input features vector. The output of the one-class classifier is more trivial and represents a Boolean YES/NO (if the sample belongs to the "base" class or not) [6].

In an offline scenario there is a training phase that precedes the classification phase [12]. During the training phase correct labels are provided by an expert or another system. In an online scenario the correct labels may be provided one by one or all at once [12], e.g. after each flight of an aircraft if there was no any fault all samples are marked as “normal”. Part of the training samples may be negative (knowingly “abnormal”). In the proposed approach OSA we use not only “normal” samples in the recursive one-class SVM, but also the “abnormal”/“negative” ones. This is the main difference with the traditional approach.

The remainder of the paper is organised as follows: in the next section the problem formulation is provided, then, in section III, the traditional SVM method is described as well as the incremental learning procedure and in section IV – the proposed OSA approach is derived and detailed. Then, in section V we validate the proposed OSA algorithm with several benchmark data sets and compare the results of this algorithm with similar algorithms available in the public domain. In section VI, the results of the performance of the algorithm are presented and analyzed. Finally, in the Conclusion, we discuss the directions of the further investigations.

2 Problem Formulation

In this paper a new approach for a single class SVM is proposed which is particularly suitable for (without being limited exclusively to) the well known problem of fault detection (FD). In supervised classification problems it is typical to have labels assigned to each data point/sample/vector, e.g. “faulty” or “normal”. In all FD problems it is a typical situation to have imbalanced data sets with the number of data for the “normal” behavior (so called “base” class) usually being hugely larger than the number of “abnormal” data which are rare. This is also the case of flight data analysis (FDA) and aviation industry where the safety is one of the key points. In addition, it is practically impossible to describe beforehand all possible faults.

For this reason, the problem can be formulated as a one-class classification [4] in which the set of normal operation (flight) data (with no faults registered) is considered as the “base” class. All *significant* deviations from the base class are considered as an event/fault.

The currently used one-class classification algorithms could be separated into two main groups:

– Support Vector Machine (SVM) -based one-class classification;
– One-class classification by the means of distribution density (membership) restoration.

In this paper, the first group of methods will be investigated and tested on real flight data from various aircraft producers.

Another important characteristic of the approach is its applicability in real time, online adaptation mechanisms in regards to its parameters to reflect the dynamically changing data pattern. In the literature there are reported incremental versions of the SVM approaches [4,5,9] including one class SVM [4,9].

All supervised learning algorithms have two stages of operation [12,13]: i) training, and ii) classification or prediction. However, it is usually impossible to collect the (flight) data, containing all possible operation modes. For this reason, it is not acceptable to use the algorithms, which can not be adapted to the new data sets after initial training. Ideally, it should be possible to adapt/evolve the classifier (fault detector) itself in terms of its structure and mechanism to fit the newly obtained data pattern especially if the changes are of a structural nature [12].

In this paper we propose a new approach which is of the one-class incremental SVM type [9]. The main innovation we propose is to use negative data samples/vectors during the training. The standard representation of the data, processed by machine learning algorithms is in the form of multidimensional vectors [13] where the dimensionality is equal to the number of features (usually, physical variables such as air speed, pitch angle, altitude, etc.).

In mathematical terms the data vector is represented as a set $X = \{x_1, \ldots, x_T\}$, where T is the duration of the whole period of observation. A subset of the observed data is supposed to be used as training set although in incremental algorithms this training data set is updated by appending it with the new data vectors [12]. In an extreme case, some classifiers can start 'from scratch with the very first data vector used as a model of the normality [2,12], but the classification rate (and the precision, respectively) will only reach more meaningful and useful values accumulating a larger number of data vectors. So, in this respect we assume that the initial number of training data set, especially when we describe a FDA system is more substantial.

For each time interval when a fault is detected and confirmed a label of "abnormal"/"faulty" can be added in real time (or as soon as the confirmation is available, to be more precise).

The following quality measures are usually used to compare different fault detection and identification methods:

– False positives (FP). This is defined as the ratio of the data vectors that are marked as faulty by the method being interrogated to the total number of non-faulty vectors, but in reality there is no fault or event.
– False negatives (FN). This is defined as a ratio of the data vectors that are supposed not to be faulty by the method that is being interrogated to the total number of faulty vectors, but an event actually took place.
– Total error (TE). This is the ratio of data vectors in which the identification result of the method being interrogated differs from the reality.

In this study a legacy FDA system used in Russia called SKAT implementing a method called express analysis, EA which is based on a very large number of thresholds and expert rules [15] is used as 'ground truth. The use of experts is very expensive and this assumptions means that in some cases a fault may have taken place but not being noticed by the experts or vice versa. Therefore, the comparison is with EA (used in SKAT) and not necessarily with the reality.

3 Method Description

The main idea of the popular SVM method is to solve the classification problem by separating the data vectors in two groups by a hyper-plane [1]. This hyper-plane is selected in such a way to maximize the "gap" between the classes. In the case of linearly separable data vectors/samples it is possible to map the data into a higher dimensionality space and to solve the classification problem there. This technique is known as the "kernel trick" [12]:

$$\begin{cases} \frac{1}{2}\|w\|^2 + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i - \rho \to \min\limits_{w,\xi,\rho} , \\ \langle x_i, w\rangle - \rho \geqslant -\xi_i,\ i = 1,\dots,m , \\ \xi_i \geqslant 0 , \end{cases} \tag{1}$$

where $X = x_1, \dots, x_m$ denotes the sample data set;
w is the normal vector;
ξ measures the degree of misclassification;
ρ is a free term of the hyper-plane;
ν is the upper bound on the part of non-marginal support vetcors and a lower limit on the fraction of support vectors.

The kernel is assumed to satisfy the Mercer's conditions [4,11].

The easiest way to solve the main optimization problem is through the dual problem:

$$\begin{cases} W = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\langle x_i, x_j\rangle + \rho(1 - \sum_{i=1}^{m}\alpha_i) \to \min\limits_{w,\xi,\rho} , \\ 0 \leqslant \alpha_i \leqslant \frac{1}{\nu m},\ i = 1,\dots,m , \end{cases} \tag{2}$$

where α is the Lagrangian multiplier [7]. This property was proven in [9], and is very useful in most of the practical cases as it gives information about the asymptotic FP rate in one-class SVM methods. The optimal hyper-plane parameters that are solutions of (1) or, equivalently, (2) can be obtained as a linear combination of the data:

$$w = \sum_{i=1}^{m}\alpha_i x_i . \tag{3}$$

All points which are used for algorithm training and for which $f(x_i) \leqslant 0$ are called support vectors (SV). Here $f(x) = sign(\langle x, w\rangle - \rho) = sign(\sum\limits_{i=1}^{m}\alpha_i\langle x_i, x\rangle - \rho)$.

The proposed learning model is one-class SVM with a Gaussian kernel:

$$f(x) = sign(\sum_{i=1}^{m} \alpha_i \exp(-\gamma \|x_i - x\|^2) - \rho) \ . \tag{4}$$

One of the drawbacks of the proposed model is a problem to select optimal parameter combination of ν and γ.

The optimization problem (2) is a quadratic one and it has a numerical solution which is described by so called *Karush-Kuhn-Tucker* (KKT) conditions [8], which provide both necessary and satisfactory conditions for the optimization problem (2) to have an optimal solution:

$$\frac{\partial W}{\partial \alpha_i} = f(x_i) \begin{cases} > 0, \, \alpha_i = 0 \\ = 0, \, 0 < \alpha_i < \dfrac{1}{\nu m} \\ < 0, \, \alpha_i = \dfrac{1}{\nu m} \ , \end{cases} \tag{5}$$

$$\frac{\partial W}{\partial \rho} = \sum_{i=1}^{m} \alpha_i - 1 = 0 \ . \tag{6}$$

It is always possible to divide the whole data set into the three separate (non-overlapping) data sub-sets as follows:

1. Subset C (correct): Data vectors which were correctly classified by the algorithm.
2. Subset M (marginal): Data vectors which are situated on the border line of the “base” class.
3. Subset E (errors) Data vectors that are misclassified by the algorithm.

It is also possible to update the SVM classification (the hyper-plane) to accommodate possible newly obtained data vectors [7]. Traditionally, SVM model is trained in an offline mode, but analytically optimal solution can also be obtained during online SVM parameters model update [9].

This method updates the SVM parameters with every new data vector, x_c by the following update rule which we provide without derivation (for more details, please, see [9]):

$$\alpha^{new} = \begin{bmatrix} \alpha_1^{old} \\ \vdots \\ \alpha_m^{old} \\ \alpha_c^{old} \end{bmatrix} + \begin{bmatrix} \Delta\alpha_1 \\ \vdots \\ \Delta\alpha_m \\ \Delta\alpha_c \end{bmatrix} \ , \tag{7}$$

$$\rho^{new} = \rho^{old} + \Delta\rho \ . \tag{8}$$

The overall idea is to update the parameters of the model – the Lagrange multipliers for *all* the previous data and set new multiplier for the newly obtained data as well as change the free term value. As a result of the changes, the solution will still satisfy the KKT conditions.

If the newly obtained data vector is correctly classified, than no update is needed, as corresponding α_c value is supposed to be 0 and KKT conditions are fulfilled. If the new data vector is misclassified by the algorithm, than it becomes a support vector (SV) and its α_c value is above 0. Initially set to 0, it is increased by a special procedure. The changes are performed in the way that the function $f(x_i)$ is still situated at 0 for all x_i in the subset M (marginal cases).

The following expression describes this in mathematical terms [9]:

$$Q_m \begin{bmatrix} -\Delta\rho \\ \Delta\alpha_M \end{bmatrix} = - \begin{bmatrix} 1 \\ k_{M,c} \end{bmatrix} \Delta\alpha_c \ , \tag{9}$$

where
$Q_M = \begin{bmatrix} 0 & 1^T \\ 1 & K_M \end{bmatrix}$;
K_M is the matrix of inner products between data vectors from the subset M;
$k_{M,c}$ — is the vector of inner products between the data vectors from the subset M and the new vector x_c;
$\Delta\alpha_M$ denotes the updates of Lagrange multipliers that correspond to the data vectors from the subset M.

$$\Delta\rho = -\beta_\rho(c)\Delta\alpha_c, \ \Delta\alpha_j = \beta_\rho(c)\Delta\alpha_c,$$

where we determine

$$\begin{bmatrix} \beta_\rho(c) \\ \beta_\rho(M) \end{bmatrix} = -Q_M^{-1} \begin{bmatrix} 1 \\ k_{M,c} \end{bmatrix} \ . \tag{10}$$

Substituting this result in (9) we have:

$$\Delta f_i = \tau_i(c)\Delta\alpha_c, \ \forall i \in \{1, \ldots, m\} \bigcup \{c\} \ ,$$

where

$$\tau_i(c) = \langle x_i, x_c \rangle + \sum_{m \in M(\alpha)} k(x_i, x_m)\beta_m(c) + \beta_\rho(c), \ \forall i \notin M(\alpha) \ ,$$

$$\tau_i = 0, \ \text{if } i \in M(\alpha) \ .$$

The increase of α_c proceeds until one of the following conditions is satisfied.

1. For one of the data vectors in the subset C the function $f(x)$ obtains value 0. Then this data vector/point is moved to the subset M with α set to 0 and α_c is increased with the new structure of the data subsets.
2. For one of the data vectors in the subset E the function $f(x)$ obtains value 0. Then this data vector/point is moved to the subset M with α set to 0 and α_c is increased with the new structure of the data subsets.
3. For one of the data vectors in the subset M the value of α gets value $\alpha = \frac{1}{\nu m}$. Then this data vector/point is moved to the subset E with α set to 0 and α_c is increased with the new structure of the data subsets.

4. For one of the data vectors in the subset M the value of α gets value 0. Then this data vector/point is moved to the subset C with α_c is increased with the new structure of the data subsets.
5. $\alpha_c = \frac{1}{\nu m}$. Then the new data vector is moved to the subset E and the algorithm terminates.
6. $f(x_c) = 0$. Then the new data vector is moved to data subset M with the current α_c and the algorithm terminates.

Due to the fact, that in the cases 1)-4) the structure of the sets E or/and M is changed, the corresponding inner product matrices should be re-computed. These operations could be speed-up by Woodberry matrix inverse formulas and effective array management.

4 The Proposed Method OSA

If consider one-class classification techniques, they usually fit their own parameters in the way to reduce the FP rate (false alarms) primarily [3,4,6,9]. At the same time, the FN rate (misses of real anomalies) is assumed to be low due to the robust model selection (using a generic type of the classification model). But in practice this approach causes high negative reactions, so the data vectors/points that do not belong to the "base" class are classified incorrectly. In case of flight data processing using statistical approaches one of the most significant problems is the high FN rate (misses).

On the other hand, two-class classification methods [10] are not applicable for fault identification problem. These approaches can identify the "fault" class more precisely, but only the cases, represented in the training data set. However, in reality it is impossible to collect (flight) data, in which all possible faults are represented.

For this reason in the proposed approach OSA we take into account the fault examples as well. The parameters of the classifier are fitted in such a way that the SVM model can take into account not only the FP rate, but also the FN rate. At the same time, the model structure is still one-class classification-type incremental SVM.

The proposed modification concerns the regularization weights which are different and are applied for FP and FN errors separately. We assume the existence of two classes which are labeled by 1 and -1, where 1 corresponds to the "base" class ((flight) data with no faults), and -1 – to the "faulty" class. However, the problem is still one-class classification type.

In mathematical terms, the proposed OSA method can be formulated as the following optimization problem:

$$\begin{cases} \frac{1}{2}\|w\|^2 + \frac{1}{\nu_1 m_1}\sum_{i=1}^{m_1}\xi_i[y_i = 1] + \frac{1}{\nu_2 m_2}\sum_{i=1}^{m_2}\xi_i[y_i = -1] - \rho \to \min \,, \\ \qquad y_i(\langle x_i, w\rangle - \rho) \geqslant -\xi_i,\ i = 1,\ldots,m \,, \\ \qquad \xi_i \geqslant 0,\ i = 1,\ldots,m \,, \end{cases} \tag{11}$$

where m_1 is the number of positive examples in the learning sample, m_2 is the number of negative examples. It can be regarded as one-class version of the cost-sensitive SVM algorithm, described at [19].

Using standard Lagrange multipliers approach, the corresponding dual optimization problem can be derived. The resulting dual optimization is represented as follows:

$$\begin{gathered} -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\langle x_i, x_j\rangle - \rho(1 - \sum_{i=1}^{m}\alpha_i y_i) \to \max \\ w.r.t. 0 < \alpha_i < \frac{1}{\nu_1 m_1}[y_i = 1] + \frac{1}{\nu_2 m_2}[y_i = -1] \;. \end{gathered} \tag{12}$$

Here α_i is a Lagrange multiplier that corresponds to correctness of data classification limitation. Resulting classification rule is as follows:

$$sign(\sum_{i=1}^{m}\alpha_i y_i\langle x_i, x\rangle - \rho) \;.$$

One can see that minor changes in the initial dual problem formulation enable us to take into account examples of the "negative" ("faulty") class with different regularization weights. Thus, in this work it is proved that cost-sensitive modification of one-class SVM with negative examples can be reduced to the standard unbalanced nu-SVM [11]. Using the same approach as it was described earlier; we propose an algorithm for the incremental update of the SVM model to accommodate the newly obtained data vectors. The algorithm is otherwise similar to the incremental SVM in respect to the parameters update, but upper margin value is selected taking the corresponding class label into account. The resulting algoritm can be represented as follows.

```
Algoritm OSA
Input: SVM model $\alpha^{old}$, $\rho^{old}$, data vector $x_c$ and its label $l_c$

1. Evaluate $f(x)$
2. If $f(x) > 0$, no model update is needed, $\alpha_c := 0$
3. Else set $\alpha_c := 0$
4. Until one of conditions 1)-6) is satisfied
   (a) Compute $\Delta\alpha_c$, $\Delta\alpha_M$ and $\Delta\rho$ using (7)-(9), taking into
       account (11)
   (b) Update C, M and E sets

Output: SVM model $\alpha^{new}$, $\rho^{new}$
```

5 Tests on Benchmark Data Sets

In this section results of algorithm validation on a number of benchmark data sets are represented. As one-class classification data sets are not widely represented, the tests are performed over standard classification data sets, one of the classes was selected as a base class with all the remaining classes considered as outliers.

The following tests were performed. The data were split into two subgroups: training and testing subgroup respectively for each data set. The algorithm was then trained using on the training subset and the performance was measured on the testing subset. A Gaussian kernel function was selected with its parameter γ determined by a cross-validation selecting the point with lower error in terms of FN and FP as illustrated in Figures 1-2 using in effect a Pareto multicriteria optimization; ν was fixed at 0.05, so the FP on the learning set were limited. The error rates are provided in the Table 1.

Table 1. Total error on benchmark data sets, in %

Data set	SVM	eClass0	eClass1	C4.5	kNN	OSA
Pima	31.03	30.62	**23.36**	26.37	25.55	26.33
Iris	10.52					10.53
Wine	5.77	7.56	**2.78**	7.87	3.06	5.77

One can see, that the algorithms performance is quite similar on the standard datasets like Wine and Iris, which are assumed to be well-separable. However, the OSA results on Pima dataset are slightly better, than the standard SVM approach produces.

6 Tests on Real Flight Data

In this study Boeing 737-800 real flight data was used. The data was collected from 138 real flights, represented in a binary format. Flight data was preprocessed, in order to represent every flight as a multidimensional time series. For this reason all parameters were re-sampled using a linear interpolation technique resulting in some frequency time series. Then, a preliminary FDA analysis was performed over the training data in order to detect the faults using flight data and to be able to compare current FDA systems with the proposed one. Most of the events, detected by the legacy systems are different kinds of pilot errors, which are hardly separable in the data space.

Cross-validation was performed in order to identify the dependence of different error rates from parameters and of the SVM algorithm. For this case a special flight data subset was selected. It was formed by data vectors, registered during flight phases “approach” and “final approach”. This set was split into two: “training” and “testing” subsets. Training procedures were performed on the first one; the second one was again used to identify the quality of the algorithm.. Further, for each pair of and values the algorithm was fitted on “training set” and the quality was verified on “testing set”.

The FP and FN rates of the proposed algorithm on test data set as a function of and is depicted in Figures 1 and 2.

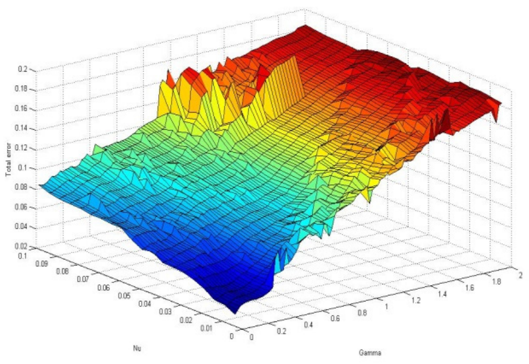


Fig. 1. FP rate as a function of ν and γ

Figure 1 represents the total error of the proposed algorithm OSA as a dependence on the model parameters ν and γ. The axis Nu that represents the ν-values; $Gamma$-axis represents γ-values.

The same dependence, but for FN rate of the proposed algorithm is depicted in the Figure 2.

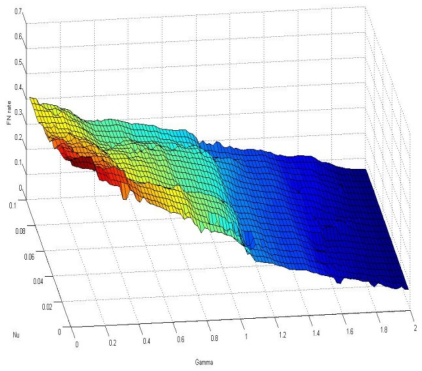


Fig. 2. FN rate as a function of ν and γ

As it can be seen from the plots in Figures 1 and 2, the dependence of the errors from the algorithm parameters is close to linear. The dependences of FP and FN from ν and γ is inverse, so any decrease of FN rate could be performed only with additional FP rate growth. The impact of the change of the γ-parameter is stronger in comparison to the ν-value.

The parameter selection was performed as follows: the maximum acceptable FP level was defined by flight safety experts. Then, all other algorithm parameters were adopted in order to decrease the FN rate with limited FP rate.

In order to have a comparison between standard SVM approach [5] and the proposed OSA approach over the full available data set twenty flights were selected as a "training" set, so both SVM models were fitted over that data. The tests of the algorithm quality were applied over the 118 flights left.

The following parameters were used while testing:

$$\nu_1 = 2,\ \nu_2 = 0.1,\ \gamma = 0.3\ .$$

Application of the incremental SVM approach is too hard computationally, as the vector sets (Marginal, Error, Correct) are increasing at every iteration of the algorithm. In order to reduce the computational complexity, the size of the set, containing correctly classified samples is limited in size (1000 vectors were selected for this purpose). In this way the model, fitted over several million vectors contains only 3000 support vectors, which is acceptable for the available computational resources.

The method performance is represented as ROC-curves in Figures 3 and 4. The curve is built by varying of the threshold parameter ρ of each of the models. Obtained plots are represented below.

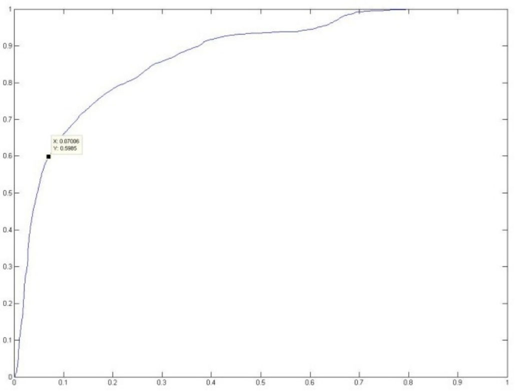


Fig. 3. ROC-curve *SVM*

The horizontal axis represents the FP rate; the vertical axis represents the true-negative rate, which can be calculated as $1 - \text{FN}$. The same curve for the proposed OSA method is represented in Figure 4.

Comparing the two figures, one can see, that the results for the modified one-class SVM are significantly better in comparison with the standard approach.

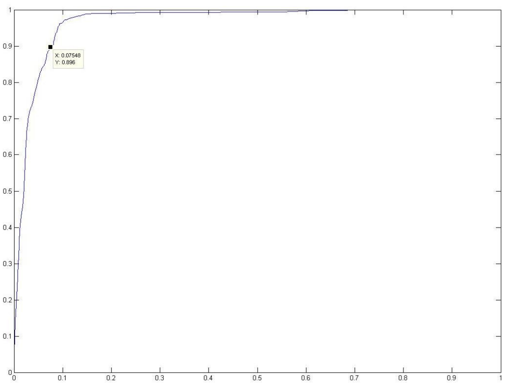


Fig. 4. ROC-curve *OSA*

A 5% limitation for the standard SVM approach allows to achieve 60% of correct alarms, therefore 40% level of FN is obtained. The proposed approach OSA allows for 5% FP limitation to get 85% of true-negative rate, therefore 15% of FN.

7 Conclusion

In this paper a new type of incremental, online SVM approach (OSA) is proposed. It builds upon recently reported online recursive one-class SVM approach [9] by generalizing to also include in considerations the "negative" (faulty) data samples as well as the "normal" ones, but is still one-class classification type. It was tested on a number of benchmark data sets as well as on real flight data. Cross-validation procedure was used to identify the parameters of the SVM algorithm. The results demonstrate improvement and are acceptable for this type of problems. In the future work the main task will be to reduce the FN rate. One of the perspective approaches to achieve this aim is to use autonomously learning classifiers such as eClass [2,12,18].

Aknowledgements. The research led in this project has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° ACPO-GA-2010-265940, "SVETLANA". Authors would like to express gratitude to Dr. Pavel Laskov for his help at the initial stage of algorithm investigation.

References

1. Vapnik, V., Golovich, S.E., Smola, A.: Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing (1997)
2. Angelov, P.: Anomalous System State Identification. GB1208542.9, priority date (May 15, 2012)

3. Koppel, M., Schler, J.: Authorship Verification as a One-Class Classification Problem. Dept. of Computer Science Bar-Ilan University Ramat-Gan, Israel
4. Gretton, A., Desobry, F.: On-line one-class support vector machines an application to signal segmentation. In: Proc. IEEE ICASSP, Hong Kong (2003)
5. Gâlmeanu, H., Andonie, R.: Implementation Issues of an Incremental and Decremental SVM. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 325–335. Springer, Heidelberg (2008)
6. Das, S., Oza, N.C.: Sparse Solutions for Single Class SVMs: A Bi-Criterion Approach. In: Proc. SDM, SIAM, pp. 816–827. Omnipress (2011)
7. Jensen, S.: An Introduction to Lagrange Multipliers, `http://www.slimy.com/~steuard/teaching/tutorials/Lagrange.html`
8. Gershwin, S.B.: KKT — Examples. MIT Open Course Ware (2010)
9. Tax, D.M.J., Laskov, P.: Online SVM learning: from classification to data description and back. Journal of Machine Learning Research 7, 1909–1936 (2006)
10. Hill, S.I., Doucet, A.: Adapting Two-Class Support Vector Classification Methods to Many Class Problems. In: Proc. of the 22nd Intern. Conf. on Machine Learning, ICML 2005, pp. 313–320 (2005)
11. Scholkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high dimensional distribution. Neural Computation 13(7), 1443–1471 (2001)
12. Angelov, P.: Autonomous Learning Systems: From Data Streams to Knowledge in Real time. Willey (December 2012) ISBN: 978-1-1199-5152-0
13. Bishop, C.: Machine Learning and Pattern Classification, 2nd edn. Springer (2009)
14. Stibor, T., Timmis, J., Eckert, C.: A Comparative study of real-valued negative selection to statistical anomaly detection techniques. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 262–275. Springer, Heidelberg (2005)
15. TU-204-100 Aircraft Operations Manual. Tupolev ANTK (1998)
16. Analysis Ground Station (AGS), Sagem/SAFRAN, France, `http://www.sagem-ds.com/spip.php?rubrique230` (accessed November 8, 2012)
17. Kolev, D., Zvikhachevskiy, D., Angelov, P.: Safety (and maintenance) improvement Through automated flight data analysis, Scale Focused Research Project for project SVETLANA, Grant Agreement: ACPO-GA-2010-265940 (March 19, 2012)
18. Angelov, P., Zhou, X.: Evolving Fuzzy-Rule-based Classifiers from Data Streams. IEEE Trans. on Fuzzy Systems 16(6), 1462–1475 (2008)
19. Brefeld, U., Geibel, P., Wysotzki, F.: Support vector machines with example dependent costs. In: Proceedings of the European Conference on Machine Learning (2003)

EEG Dataset Reduction and Classification Using Wave Atom Transform

Ignas Martisius, Darius Birvinskas, Robertas Damasevicius, and Vacius Jusas

Kaunas University of Technology, Software Engineering Department,
Studentu st. 50, Kaunas, Lithuania

Abstract. Brain Computer Interface (BCI) systems perform intensive processing of the electroencephalogram (EEG) data in order to form control signals for external electronic devices or virtual objects. The main task of a BCI system is to correctly detect and classify mental states in the EEG data. The efficiency (accuracy and speed) of a BCI system depends upon the feature dimensionality of the EEG signal and the number of mental states required for control. Feature reduction can help improve system learning speed and, in some cases, classification accuracy. Here we consider Wave Atom Transform (WAT) of the EEG data as a feature reduction method. WAT takes input data and concentrates its energy in a few transform coefficients. WAT is used as a data preprocessing step for feature extraction. We use artificial neural networks (ANNs) for classification and perform research with varying number of neurons in a hidden layer and different network training functions (Levenberg-Marquardt, Conjugate Gradient Backpropagation, Bayesian Regularization). The novelty of the paper is the application of WAT in the EEG data processing. We conclude that the method can be successfully used for feature extraction and dataset feature reduction in the BCI domain.

Keywords: EEG, Brain Computer Interface, Wave Atom Transform, dimensionality reduction, classification.

1 Introduction

The EEG data is inherently complex. The signals are non-stationary, non-linear, and thus difficult to analyse. Classifying EEG data requires, firstly, the reduction of its high-dimensional feature space to identify fewer intrinsic feature dimensions relevant to specific mental states of a subject. To perform classification, Artificial Neural Network (ANN) is a good choice due to its ability to generalize and work well with noisy data [1]. However, long training time is a well-known disadvantage of ANN, especially when a BCI system must deal with real-time constraints [2]. Training time is important because the system classifier needs to be retrained constantly in a real time BCI application. The subject's mental state changes in time, and the classifier must adapt to these changes. Also training must be performed according to the control task at hand, i.e. different video game controls require differently trained classifiers.

© Springer-Verlag Berlin Heidelberg 2013

Presenting a network with a subset of a spectrogram as an input vector is the obvious solution. Such a system has been used effectively to classify EEG data [3]. This technique, however, still has a high computational cost since large numbers of weights must be updated in training. Classical signal transforms (such as FFT or DCT) decompose time series into global sinusoidal components of fixed amplitude. However, the Fourier methods may not give an efficient representation of a signal. Wavelet transforms have uniform temporal resolution for all frequency scales, but resolution is limited by the basic wavelet [4]. These transforms are useful for characterizing gradual frequency changes.

Wave Atom Transform (WAT) has been recently proposed by Demanet and Ying [4]. WAT performs a multi-resolutional analysis of a signal, i.e., decomposes a signal into different frequency sub-bands. Wave atoms are a variant of wavelets that have a sharp frequency localization and offer a sparser expansion for oscillatory functions than wavelets. Wave atoms compose wave-fields as a superposition of highly anisotropic, localized, and multi-scale waveforms and capture coherence of pattern across and along oscillations. WAT has been previously used mainly in image processing for image denoising [5], image watermarking, image hashing, fingerprint recognition [6], signal analysis [7], as well as feature extraction [6], dimensionality reduction and numerical analysis [5]. To our knowledge, wave atoms have not been applied to EEG signal processing. Similar research included ECG (electrocardiogram) [8] and MRI (Magnetic Resonance Image) [9] data. WAT is a promising approach for EEG processing because of its denoising and feature extraction capabilities, and is particularly useful when the signal has discontinuities and sharp spikes as is in case of EEG. We expect that WAT coefficients extracted from EEG data samples can retain enough information to permit correct classification, while feature reduction should reduce network training and classification time.

We analyse and perform experimental research on the ANN training functions to find the best combination of training speed and classification accuracy as applied to the WAT coefficients of the EEG data.

The remaining parts of the paper are as follows. Section 2 provides a short introduction into WAT. Section 3 overviews the ANNs and their training functions. Section 4 describes a case study using an openly available dataset and discusses experimental results. Finally, Section 5 presents the conclusions.

2 Wave Atom Transform

Wave atoms are a variant of 2D wavelet packets that retain an isotropic aspect ratio. They are well suited for representing oscillatory patterns in a signal [5]. A signal is regarded as an oscillatory pattern if it can be described as (1)

$$f(x) = sin(Ng(x))h(x) \ , \tag{1}$$

where x is a coordinate, g and h are the C^{∞} scale function, N is a large constant. WAT allows to obtain a sparser solution of the oscillatory pattern signal f than that of other known transformations, as only $O(N)$ wave atom coefficients are sufficient to represent f to some given accuracy [6], while $O(N^2)$ wavelet coefficients are needed in the same case.

The definition of wave atoms for 1D signal is as follows. Consider $f(x)$ and $\hat{f}(w)$ is a 1D Fourier transform pair, x, w corresponds to the coordinates in time domain and frequency domain. Define wave atoms as $\phi_\mu(x)$, in which $\mu = (j; m; n)$. Then the indexed point $(x_\mu; w_\mu)$ in phase-space is defined in (2).

$$x_\mu = 2^{-j}m \; ; \; \omega_\mu = \pi 2^j n \ . \tag{2}$$

The definition is similar to that of wavelet packets, where j controls the resolution scale, m and n control the location in time and frequency domain.

The elements of frame ϕ_μ are called wave atoms when:

$$\begin{aligned} |\hat{\phi}_\mu(\omega)| \leq & \; C_M 2^{-j}(1 + 2^{-j}|\omega - \omega_\mu|)^{-M} + \\ & + C_M 2^{-j}(1 + 2^{-j}|\omega + \omega_\mu|)^{-M} \ ; \\ |\phi_\mu(x)| \leq & \quad C_M 2^j (1 + 2^j|x - x_\mu|)^{-M} \ . \end{aligned} \tag{3}$$

for all $M > 0$. The difference between wavelet packets and wave atoms are that wave atoms obey the parabolic scaling wavelength: at scale 2^{-2j} the essential frequency support is of size 2^j while at frequency 2^{2j}, the essential time support is of size 2^j [5]. Also, the basis function is different from a conventional wavelet basis function. The frequency domain formula of a basis function is defined as (4).

$$\hat{\Psi}^0_m(\omega) = e^{-i\omega/2}[e^{i\alpha_m} g(\epsilon_m(\omega - \pi\kappa)) + e^{-i\alpha_m} g(\epsilon_{m+1}(\omega + \pi\kappa))] \ . \tag{4}$$

where $\kappa = \left(m + \frac{1}{2}\right)$, $\epsilon_m = (-1)m$ and $\alpha_m = \left(\frac{\pi}{2} + \frac{1}{2}\right)$. The function g is an appropriate real-valued function, compactly supported on an interval of length 2π, and chosen according to (5).

$$\sum_m |\hat{\Psi}^0_m(\omega)|^2 = 1 \ . \tag{5}$$

3 ANN and Training Functions

ANN is a mathematical model that tries to mimic functional aspects of a biological neuron network. The ANN is comprised of interconnecting artificial neurons. These neurons, sometimes called nodes or units, perform simple calculations. The neuron receives input from other neurons or from external sources. Each neuron has an associated weight, which is modified to simulate learning. A single neuron is a relatively simple computational element, so a network of these neurons is most often used for complex tasks. Two or more neurons can be combined to form a layer. Each input of a layer is connected to inputs of the next layer through neuron weights. Most ANNs have an input layer, an internal, or hidden layer and an output layer.

The training of ANN's is a numerical optimization of a nonlinear error function. There is no single best method for solving this problem. The training method must be chosen according to the characteristics of the problem to be solved and the ANN configuration itself. For our experiment 3 different training functions were chosen.

Levenberg-Marquardt Training Function (LM) provides a numerical solution to the problem of minimizing a non-linear function. It is fast and has stable convergence. In the artificial neural-networks field, this algorithm is suitable for training small- and medium-sized problems.The algorithm is an iterative technique that locates the minimum of a multivariate function that is expressed as the sum of squares of non-linear real-valued functions.

Fletcher-Powell Conjugate Gradient Backpropagation Training (CGF) makes a search along the conjugate gradient direction to determine the step size, which minimizes the performance function along that line. The algorithm starts by searching in the steepest descent direction. A line search is then performed to determine the optimal distance to move along the current search direction, then the next search direction is determined so that it is conjugate to previous search directions. The line search avoids the need to compute the Hessian matrix of second derivatives, but it requires computing the error at multiple points along the line. The general procedure is to combine the new steepest descent direction with the previous search direction.

Bayesian Regularization Training Function (BR) updates the weight and bias values according to Levenberg-Marquardt optimization. It minimizes a linear combination of squared errors and weights, and then determines the correct combination so as to produce a network that generalizes well. The weights and biases of the network are assumed to random variables with specified distributions, with prior and posterior probabilities.

4 Case Study

Data set Ia, provided by the University of Tübingen, from the BBCI competition II (`http://www.bbci.de/competition/`) was used. Dataset contains measurements of slow cortical potentials recorded in 6 channels (sampling rate 256 Hz, recording length 3.5 s) and interpreted as follows. Cortical positivity means a downward movement, and cortical negativity means an upward movement of the cursor on the screen [12]. This data set consists of 268 training trials (135 trials belongs to class 0 and 133 trials belongs to class 1) and 293 testing trials. A separate testing and training dataset allows for more reliable results.

Classification was performed using strictly feed-forward ANNs with one input layer, one output layer and one hidden neuron layer, initialized with random values. A tangent sigmoid threshold function was used both in hidden and output layers. A 15-fold cross validation was performed for every ANN hidden layer size

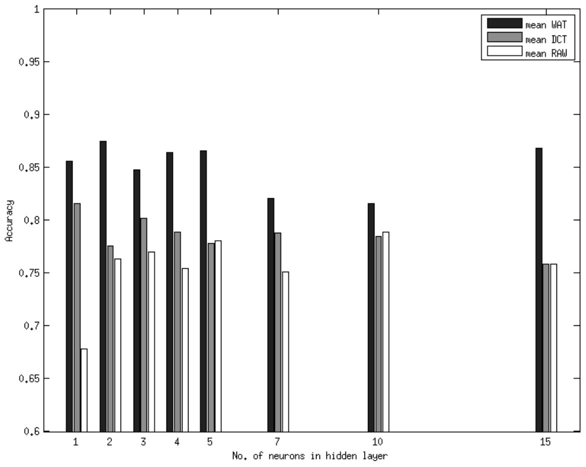


Fig. 1. Accuracy using Levenberg-Marquardt training of ANN

and average accuracy, F-measure and training time was calculated. For comparison, raw EEG data, 50 DCT coefficients [3] and WAT 1st scale coefficients were used as inputs.

In this paper classification accuracy was measured as a ratio of true results and the total number of trials in percent. This helps to compare results to the BBCI competition II. To further evaluate classification accuracy, *F-measure* (a harmonic mean of precision and recall) was used. Training speed was evaluated by an average time needed for one ANN training. A PC running Debian Linux with an AMD Phenom II X4 (3.6 GHz) processor and 8GB RAM was used. Classification accuracy is presented in Fig.1-3. Training time comparison is presented in Table 1.

Classification results using Levenberg-Marquardt training are shown in Fig. 1. Accuracy of 88% was achieved using 2 neurons in the hidden layer, with an average training time of 0.49 s. WAT transform showed better results than DCT or RAW with all network configurations.

As Fig. 2 shows, using Fletcher-Powell Conjugate Gradient Backpropagation (CGF) training produced similar accuracy for all hidden layer sizes. Best classification result for WAT transform was an accuracy of 88%. This was achieved with 1 neuron in the hidden layer with a network training time of 0.79 s. The same accuracy (88%) was achieved with a hidden layer size of 5 neurons and DCT transform (training time was 0.56 s).

Using Bayesian Regularization training showed best results in classification accuracy (Fig. 3) and the best training speed improvement overall. Best classification result 90% was achieved with 2 neurons in the hidden layer with a training time of 1.1 s. No results for hidden layer sizes of 10 and 15 neurons for raw EEG data are presented because of high network training time.

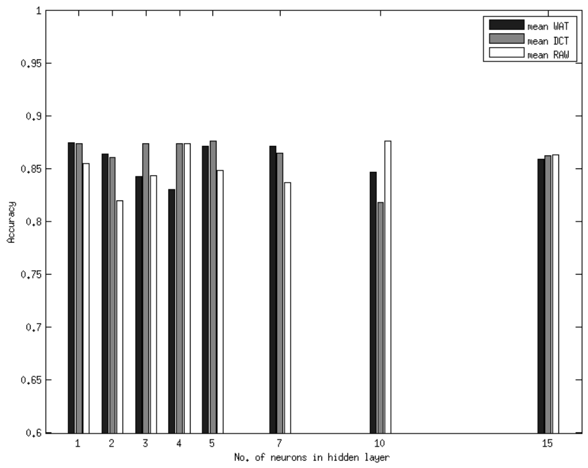


Fig. 2. Accuracy using Fletcher-Powell Conjugate Gradient Backpropagation training of ANN

Experimental results are summarized in Table 1. Average training time and classification accuracy for a neural network achieving best results are presented. Training time is dramatically decreased with the use of a reduced training dataset. WAT shows better training time and classification results using Levenberg-Marquardt and Bayesian Regularization functions. The box chart for F-measure results is presented in Fig. 4. Therefore, WAT has a great potential for application in real-time BCI tasks.

As shown in Table 1 classification using WAT transform is most accurate with all training functions. All classification quality results are in line with the best results obtained in the BBCI competition II. Best accuracy of 90% is achieved using Bayesian Regularization training, however it is the slowest. These results show that the use of the Levenberg-Marquardt training reduces ANN training time by half, with an negligible accuracy loss.

Table 1. Comparison of classification accuracy and network training time

Training function	Features	Neurons	Accuracy, %	F-measure	Time, s
Levenberg-Marquardt	RAW	10	79	0.78	234.4
	DCT	1	82	0.80	0.56
	WAT	2	87	0.88	**0.49**
Fletcher-Powell Conjugate Gradient Backpropagation	RAW	10	87	0.87	0.93
	DCT	5	88	0.88	0.56
	WAT	1	88	0.88	0.79
Bayesian Regularization	RAW	5	84	0.84	11906
	DCT	15	85	0.85	40
	WAT	2	**90**	**0.90**	**1.1**

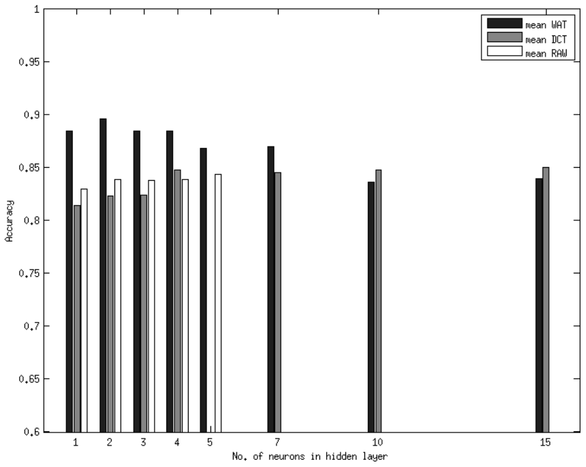


Fig. 3. Accuracy using Bayesian regularization training of ANN

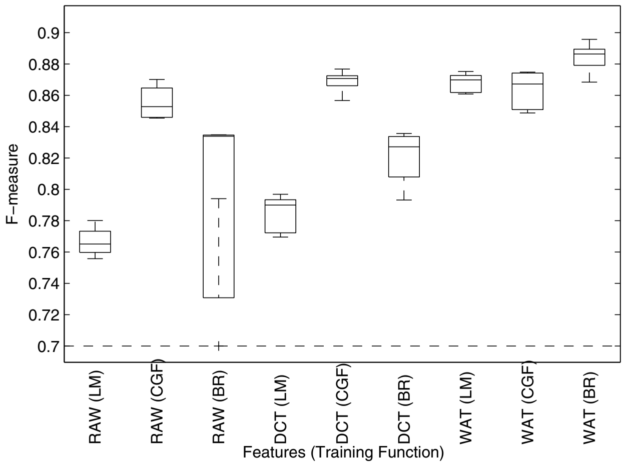


Fig. 4. F-measure box diagram

5 Conclusions

We propose the use of Wave Atom Transform (WAT) for EEG feature extraction and dataset reduction. Our results show this method to be effective for reducing data size without the loss of important signal information. Data classification was performed using artificial neural networks with various hidden layer sizes and training functions. Results show a dramatic improvement of training speed with a transformed dataset. The best improvement was achieved using Bayesian regularization training. This improvement would mostly benefit real-time BCI applications.

Future research will include the use of WAT on self-recorded data for use in a personal portable BCI system rather than publicly available datasets.

Acknowledgments. The work described in this paper has been carried out within the framework the Operational Programme for the Development of Human Resources 2007-2013 of Lithuania Strengthening of capacities of researchers and scientists project VP1-3.1-MM-08-K-01-018 Research and development of Internet technologies and their infrastructure for smart environments of things and services (2012- 2015), funded by the European Social Fund (ESF).

References

1. Subasi, A., Ercelebi, E.: Classification of EEG signals using neural network and logistic regression. Computer Methods and Programs in Biomedicine 78(2), 87–99 (2005)
2. Martisius, I., Sidlauskas, K., Damasevicius, R.: Real-Time Training of Voted Perceptron for Classification of EEG Data. International Journal of Artificial Intelligence 10(S13), 41–50 (2013)
3. Birvinskas, D., Jusas, V., Martisius, I., Damasevicius, R.: EEG dataset reduction and feature extraction using discrete cosine transform. In: Proc. of UKSim-AMSS EMS 2012: 6th European Modelling Symposium on Mathematical Modeling and Computer Simulation, Malta, November 14-16, pp. 186–191 (2012)
4. Addison, P.A.: Wavelet transforms and the ECG: a review. Physiological Measurement 26, 155–199 (2005)
5. Demanet, L., Ying, L.: Wave atoms and sparsity of oscillatory patterns. Appl. Comput. Harmon. Anal. 23(3), 368–387 (2007)
6. Mohammed, A.A., Jonathan Wu, Q.M., Sid-Ahmed, M.A.: Application of Wave Atoms Decomposition and Extreme Learning Machine for Fingerprint Classification. In: Campilho, A., Kamel, M. (eds.) ICIAR 2010, Part II. LNCS, vol. 6112, pp. 246–255. Springer, Heidelberg (2010)
7. Herrmann, F.J., Friedlander, M.P., Yilmaz, O.: Fighting the Curse of Dimensionality: Compressive Sensing in Exploration Seismology. IEEE Signal Processing Magazine 29(3), 88–100 (2012)
8. Aggarwal, V., Patterh, M.S.: ECG Compression using Wavelet Packet, Cosine Packet and Wave Atom Transforms. Int. Journal of Electronic Engineering Research 1(3), 259–268 (2009)
9. Rajeesh, J.: Rician noise removal on MRI using wave atom transform with histogram based noise variance estimation. In: IEEE Int. Conf. on Communication Control and Computing Technologies (ICCCCT), October 7-9, pp. 531–535 (2010)
10. Geetika, D., Varun, R.: MRI Denoising Using Waveatom Shrinkage. Global Journal of Researches in Engineering 12(4) (2012)
11. Martišius, I., Damaševičius, R.: Class-adaptive denoising for EEG data classification. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS, vol. 7268, pp. 302–309. Springer, Heidelberg (2012)
12. Birbaumer, N., Flor, H., Ghanayim, N., Hinterberger, T., Iverson, I., Taub, E., Kotchoubey, B., Kbler, A., Perelmouter, J.: A Brain-Controlled Spelling Device for the Completely Paralyzed. Nature 398, 297–298

Embodied Language Understanding with a Multiple Timescale Recurrent Neural Network

Stefan Heinrich, Cornelius Weber, and Stefan Wermter

University of Hamburg, Department of Informatics, Knowledge Technology
Vogt-Kölln-Straße 30, D - 22527 Hamburg, Germany
`{heinrich,weber,wermter}@informatik.uni-hamburg.de`
`http://www.informatik.uni-hamburg.de/WTM/`

Abstract. How the human brain understands natural language and what we can learn for intelligent systems is open research. Recently, researchers claimed that language is embodied in most – if not all – sensory and sensorimotor modalities and that the brain's architecture favours the emergence of language. In this paper we investigate the characteristics of such an architecture and propose a model based on the Multiple Timescale Recurrent Neural Network, extended by embodied visual perception. We show that such an architecture can learn the meaning of utterances with respect to visual perception and that it can produce verbal utterances that correctly describe previously unknown scenes.

Keywords: Embodied Language, MTRNN, Language Acquisition.

1 Introduction

Natural language is the cognitive capability that clearly distinguishes humans from other living beings and often is called the key to intelligence. In the past researchers have contributed valuable models to explain the binding of language to experience, but also to ground language in embodied perception and action based on recent neuroscientific data and hypotheses [3,6]. In addition early models captured the fusion of language and multi-modal perceptions or aimed at bridging the gap between formal linguistics and bio-inspired systems [15,16].

However, due to the vast complexity of language, some models rely on well-understood Chomskyan formal theories, which are difficult to maintain in the light of recent neuroscientific findings, e.g. of non-infinite-recursive mechanisms and the evident involvement of various – if not all – functional areas in the human brain in language [1,2,14]. Other integrating or constructive models are constrained to single words, neglecting the temporal aspect of language [10].

In a recent study Hinoshita et al. claimed that for human language acquisition just an "appropriate" architecture is sufficient and provided a model based on a *Multiple Timescale Recurrent Neural Network* (MTRNN) [11]. They found that such a system composes language hierarchically in a self-organised way, if the architecture includes dynamic interaction of components with different characteristics, e.g. information processing on different timescales. Although the

© Springer-Verlag Berlin Heidelberg 2013

model was reproducing learned symbolic sentences quite well, generalisation was not possible, because the generation of sentences was initiated by the internal state of some neurons, which had to be trained individually for every sentence.

In this paper we incorporate embodied perception based on real world data in an MTRNN model and show that such a novel system is able to generalise to completely new situations by recomposing learned elements, and also self-organises toward the meaning of the learned verbal utterances.

2 Extended MTRNN Model

For our proposed model we employ the MTRNN to process verbal utterances over time [19], extended by several feed-forward layers to integrate embodied perceptions during the processing of utterances. The MTRNN part is composed of an *Input- and Output* layer (IO) and two context layers called *Context fast* (Cf) and *Context slow* (Cs). In general, the MTRNN is an extended *Elman Recurrent Neural Network* (ERNN) on the one hand and a special case of the *Plausibility Recurrent Neural Network* (PRNN) on the other hand [5,18]. In contrast to the ERNN, the MTRNN allows for full connectivity of neurons to all neurons of the same and of adjacent layers, and introduces a mechanism forcing neurons in the context layers to process information with different timescales. Compared with the PRNN it restricts this concept of *hysteresis* to an increasing slowness from the first to the last layer and also restricts the architecture to one horizontal set of layers. Our extension part consists of an *Embodied Input* layer (EI), an *Embodied Fusion* layer (EF), and an *Embodied Controlling* layer (EC). Fig. 1 provides an overview of our architecture.

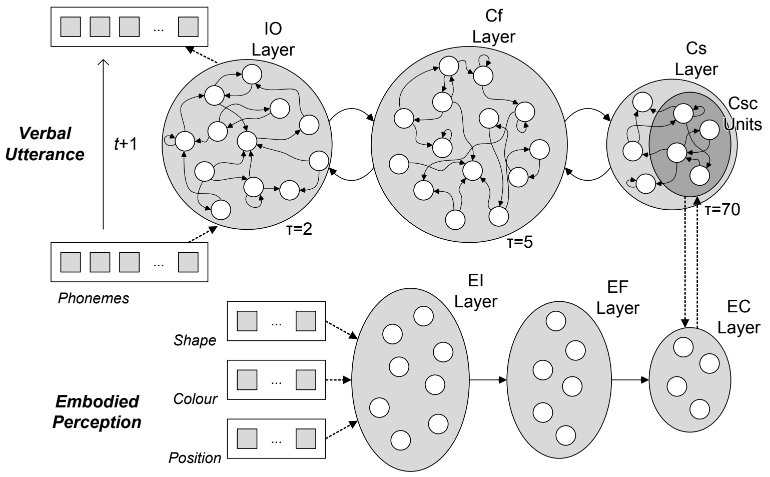


Fig. 1. Architecture of a Multiple Timescale Recurrent Neural Network extended by embodied perception from the scene. A sequence of phonemes (utterance) is processed over time, while the perceived situated information is constantly present.

During learning of the system the MTRNN is trained with verbal utterances and self-organises the neural activity and thereby the internal state values of some of the neurons in the Cs layer (so called *Context Controlling* units (Csc)). These self-organised values are then transferred to the EC layer and associated with the present embodied perception. For training we use an adaptive mechanism based on the *resilient propagation* algorithm [8]. During testing, the system approximates EC values from the visual perception input that are transferred to the Csc units, which in turn initiate the generation of a corresponding verbal utterance.

A full formal description of the MTRNN architecture can be found in the work of Yamashita and Tani [19]. In our model the MTRNN is specified by timescale values of $\tau = 2$, $\tau = 5$, and $\tau = 70$ for the IO, Cf, and Cs layers respectively, based on previous studies [11,19] and preliminary experiments (not included), which show that these settings work best for the language learning scenario. For the IO layer we employ a soft-max function, while for the neurons in the remaining layers we use the following modified logistic transfer function:

$$f(x) = \frac{1.7159}{1 + \exp(-x \cdot 0.92)} - 0.35795 \quad . \tag{1}$$

The function is modulated in slope and range to capture the characteristics of the synchronic transfer function that has been proposed by LeCun for faster convergence in association tasks [13]. As error function on the IO layer we use the *Kullback–Leibler divergence*:

$$E(W) = \sum_{t} \sum_{i \in I_{\mathrm{IO}}} d_{t,i} \cdot \log\left(\frac{d_{t,i}}{y_{t,i}}\right) \quad , \tag{2}$$

where W represents the weight matrix, y denotes the output of neuron i at time step t, and d identifies the desired activity.

3 Scenario

Our scenario for this model is the interaction between a human teacher and a robotic learner, which is supposed to learn language from scratch by grounding speech acts in its embodied experience, but also is supposed to use its learned language to describe novel situations. The robot is placed in a scene and receives an utterance from the teacher, who describes the scene, e.g. "THE APPLE HAS COLOUR GREEN". The system should learn, in a self-organised way, how to bind the visual scene information with this verbal expression to be able to describe another scene like "THE BANANA HAS COLOUR GREEN" correctly. The focus of this study is on generalisation using possibly learned components.

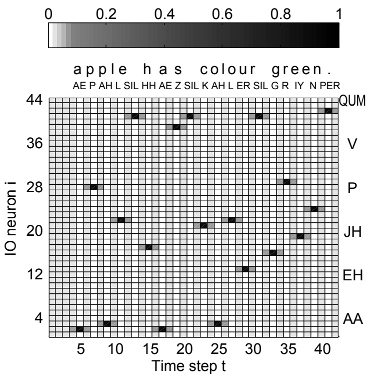
To control our setup, all verbal utterances stem from a small symbolic grammar as presented in Fig. 2a. However, every symbolic sentence is transformed into a phonetic utterance based on phonemes from the ARPAbet and four additional signs to express pauses and intonations in propositions, exclamations, and questions: $\Sigma = \{\text{'AA'}, ..., \text{'ZH'}\} \cup \{\text{'SIL'}, \text{'PER'}, \text{'EXM'}, \text{'QUM'}\}$, with size $|\Sigma| = 44$.

To encode an utterance $u = (p_1, \ldots, p_l)$ into neural activation over time, we adapted the encoding scheme suggested by Hinoshita et al. [11], but we use a phoneme-based instead of a symbol-based representation: The occurrence of a phoneme p_k is represented by a spike-like neural activity of a specific neuron at relative time step r. In addition, some activity is spread backward in time (rising phase) and some activity is spread forward in time (falling phase) represented as a Gaussian over the interval $[r - \omega/2, \ldots, r-1, r, r+1, \ldots, r + \omega/2]$. All activities of spike-like peaks are normalised by the soft-max function for every absolute time step. A detailed description can be found in [11]. For our scenario we set the constants accordingly to $\mu = 4$, $\omega = 4$, $\sigma^2 = 0.3$, and $\upsilon = 2$. The ideal neural activation for an encoded sample utterance is visualised in Fig. 2b.

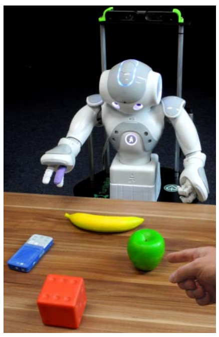
To encode the visual shape perception into sustained neural activity, we aimed at capturing the salient features of the objects in the field of view, inspired by saccadic eye movements of humans [9]. On an image taken by the NAO robot we employ the mean shift algorithm for segmentation [4], and the Canny edge detection as well as the contour finder for object discrimination. Subsequently, we calculate the centre of mass and 16 distances to salient points around the contour. Finally, we scale the distances by the square root of the object's area and order them clockwise – starting with the largest – to determine the characteristic shape, which is scale and rotation invariant. Fig. 2e provides two example results of this process and Fig. 2f visualises typical characteristics for all employed object shapes (scaled to $[0, 1]$). Encoding of the perceived colour is realised by averaging the three R, G, and B values of the shape, while the perceived position is encoded by the two values of the centroid coordinate in the field of view.

```
S → INFORM
INFORM → POS is a OBJ.
INFORM → OBJ has colour COL.
OBJ → apple | banana | dice | phone
POS → above | below | left | right
COL → blue | green | red | yellow
```

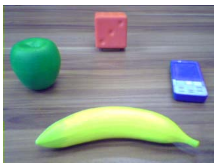
(a) Grammar.



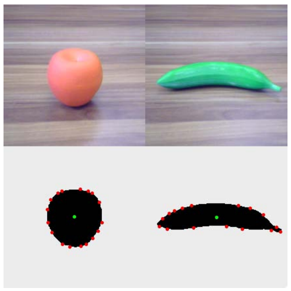
(b) Encoded utterance.



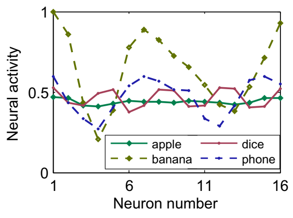
(c) Learner.



(d) Learner's view.



(e) Perceived shapes.



(f) Shape representation.

Fig. 2. Representations and scenario of language learning in human-robot interaction.

4 Evaluation and Analysis

To test and analyse our model, we collected a data set consisting of all possible scenes and their respective verbal description. From the grammar we obtained 32 different combinations, which we set up as scenes and in turn used for collecting different examples. The corresponding verbal utterances were reasonably complex sequences with a length of 32 to 46 time steps (compare Fig. 2b). Subsequently, we ran a series of experiments for which we carefully, but randomly divided the data into a training set and a test set (50:50) – making sure that every scene is included only in one of these sets – and initialised a network. For every setup we repeated this process 50 times with different random seeds. The parameters of the network were mostly chosen based on the experience in [11]: We used $|IO| = 44$ and $|EC| = 21$ constrained by the input representations, but varied the sizes of Cf, Cs and EF to test for robustness. The size of EC depends on and is equal to the size of Csc, which we determined with $|Csc| = \lceil |Cs|/2 \rceil$. In addition, we used a feedback rate $\varphi = 0.1$ and initialised the weights in the interval $[-0.025, 0.025]$ and the initial Csc in the interval $[-0.01, 0.01]$.

4.1 Generalisation

To be able to compare the generalisation capabilities, we use the standard measure F_1-score determined by precision and recall, and defined as follows:

$$p_{\text{precision}} = \frac{tp}{tp + fp}\ ,\ p_{\text{recall}} = \frac{tp}{tp + fn}\ ,\ F_1\text{-score} = 2 \cdot \frac{p_{\text{precision}} \cdot p_{\text{recall}}}{p_{\text{precision}} + p_{\text{recall}}}\ , \quad (3)$$

where we specify all correct and matching sentences as tp (true positives), all correct but not matching sentences as fp (false positives), and strictly all incorrect sentences as fn (false negatives).

The results in Tab. 1 show that the system can be trained perfectly in most cases, and also produces correct utterances for new scenes on a moderate level: For a suitable parameter setting, networks reach an F_1-score of up to 1.0 on the training set and 0.545 on the test set, with an average over all random seeds of 0.999 on the training set and 0.136 on the test set.

Table 1. Comparison of F_1-score for different network dimensions

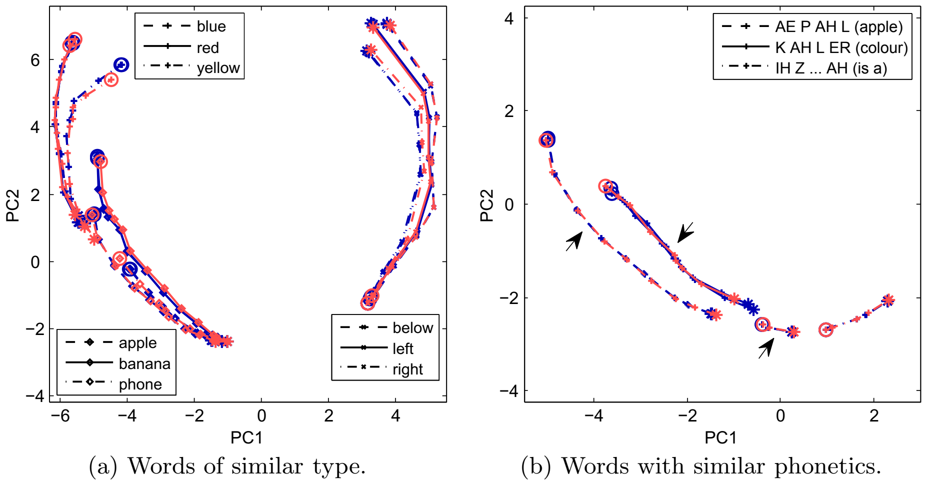
$\|Cf\|/\|Cs\|$	40/11	40/11	40/11	80/23	80/23	80/23	160/47	160/47	160/47
$\|EF\|$	8	16	24	8	16	24	8	16	24
training set best	1.000	1.000	1.000	1.000	1.000	**1.000**	1.000	1.000	1.000
test set best	0.316	0.222	0.316	0.316	0.400	**0.545**	0.316	0.476	0.400
training set average	0.889	0.904	0.908	0.950	0.999	**0.999**	0.979	0.994	0.995
test set average	0.059	0.049	0.043	0.091	0.136	**0.136**	0.096	0.091	0.123

Note that due to the random selection, in several cases the system had to describe a scene, for which it had not seen any aspect (shape, colour, or position) before. This was intended to keep the scenario realistic and observe the effects.

4.2 Network Behaviour

To provide a better understanding of the system, we analysed the neural activity of the Cf layer for the trained networks. We aimed to test, whether this layer had organised itself to represent the words in the utterances (compare [11]). Using *principle component analysis* (PCA) we reduced the dimensionality to visualise trajectories over time for specific words. The starting and the end point of the trajectory were defined as the first highest activity for the first phoneme and the last highest activity for the last phoneme of the word in the IO layer.

The results reveal several characteristics (see Fig. 3 for the trajectories of a typical network): First, the neural activity in the Cf layer is nearly identical for the same words from trained utterances. Second, the same words from untrained utterances have a quite similar activity pattern. Third, words of the same type (shape, colour, or position words) have a very related activity pattern. From the data we can observe, that the networks self-organise patterns for words about shapes, colours, and positions. Forth, words with similar phonetic representation have different activities, if the type of the word is different. Low correlation was found of activity for phonetically similar but semantically different words.



(a) Words of similar type. (b) Words with similar phonetics.

Fig. 3. Comparison of neural activation in the Cf layer for different words. The dimensionality has been reduced from $|Cf|$ to two dimensions (PC1 and PC2) and the beginning ($*$) as well as the end ($\circ$) of the words have been marked. The dark/blue lines represent words from utterances of the training set and the bright/red lines show words from utterances of the test set. Arrows indicate the same phoneme "AH".

In addition, we found the tendency that the activation of a word primes the activation of other grammatically related words. In terms of trajectories it can be observed that the end point of the word "COLOUR" is close to the starting point of all colour words, and the end point of a position word is close to the starting point of "IS A ..." (compare Fig. 3a and b).

5 Discussion

The combination of visual perception and an architecture that includes different timescales in processing verbal sequences provides a system that self-organises towards the meaning of learned utterances in a real world scenario. Our experiments have shown that such a system apparently is able to understand verbal utterances and describe novel scenes with the correct corresponding verbal utterances. The analysis revealed that novel scenes are described by recomposing the correct words, which have been grounded in the perception of different shapes, colours, or positions.

For some incorrect sentences we observed both cases: Minor substitution errors in terms of a single wrong phoneme or a pause that was too long ("SIL SIL" instead of "SIL"), as well as no meaningful phoneme chains at all. In the first case, listening humans would presumably consider this a normal inaccuracy and automatically correct the recognition. The second case clearly shows that generalisation was sometimes difficult. It is open to clarify, whether this degree of difficulty is inherent, e.g. if the error rate is comparable to certain learning stages in young children during early language learning [12].

During training of the system, we found that the connection weights from the Cf to the Cs layer as well as from the IO to the Cf layer converged towards zero in many cases. This means that the highly dynamic networks organised themselves towards a directed flow of information from the context to the phonetic output instead of a mutual exchange of information. This is plausible in the light of neuroscientific evidence [10], but for future experiments implies that the MTRNN architecture might already be more complex than necessary and should be tested with less initial connectivity. In addition we found that incorporating an adaptive training mechanism and a novel transfer function already allowed to reduce the complexity of the training itself.

Parameter exploration has shown that for this architecture a good balance of the number of neurons and the number of training samples is important. This is in line with experience from associator networks [13], but less desirable. Further investigations should include the consideration of architectures that are dynamic in connectivity as well as in size. In addition the architectures should be tested with more complex scenes and verbal descriptions, including interrelations of multiple objects and embodied experience of a broader set of real world situations.

In conclusion, our study supports that the embodiment of language in perception and a hierarchical structure with different timescales are important aspects of an appropriate architecture for language. For such an architecture a feasible constraint can be our mostly feedforward but compositional structure, also suggested for the (visual) cortex [7]. In the future we need to further refine the architectural characteristics to identify the most important building blocks for natural language processing. The understanding of the brain's architecture for language can explain the humans' most important cognitive capability, but also can inform future software frameworks for service robots that should interact with and understand humans.

Acknowledgments. The authors would like to thank A. Saleh and H. Vöcking for support with the real world data acquisition as well as J. Dávila-Chacón and S. Magg for critical and inspiring discussions.

References

1. Barsalou, L.W.: Grounded cognition. Annu. Rev. Psychol. 59, 617–645 (2008)
2. Borghi, A.M., Gianelli, C., Scorolli, C.: Sentence comprehension: effectors and goals, self and others. An overview of experiments and implications for robotics. Frontiers in Neurorobotics 4(3), 8 (2010)
3. Cangelosi, A.: Grounding language in action and perception: From cognitive agents to humanoid robots. Physics of Life Reviews 7(2), 139–151 (2010)
4. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. on Pattern Anal. and Mach. Intell. 24(5), 603–619 (2002)
5. Elman, J.L.: Finding structure in time. Cognitive Science 14(2), 179–211 (1990)
6. Frank, S.L.: Strong systematicity in sentence processing by an echo state network. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 505–514. Springer, Heidelberg (2006)
7. Friston, K.: A theory of cortical responses. Philosophical Transactions of the Royal Society B 360, 815–836 (2005)
8. Heinrich, S., Weber, C., Wermter, S.: Adaptive learning of linguistic hierarchy in a multiple timescale recurrent neural network. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 555–562. Springer, Heidelberg (2012)
9. Henderson, J.M.: Human gaze control during real-world scene perception. Trends in Cognitive Sciences 7(11), 498–504 (2003)
10. Hickok, G., Poeppel, D.: The cortical organization of speech processing. Nature Reviews Neuroscience 8(5), 393–402 (2007)
11. Hinoshita, W., Arie, H., Tani, J., Okuno, H.G., Ogata, T.: Emergence of hierarchical structure mirroring linguistic composition in a recurrent neural network. Neural Networks 24(4), 311–320 (2011)
12. Karmiloff, K., Karmiloff-Smith, A.: Pathways to language: From fetus to adolescent. Harvard University Press (2002)
13. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Orr, G.B., Müller, K.-R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, pp. 9–50. Springer, Heidelberg (1998)
14. Pulvermüller, F., Fadiga, L.: Active perception: sensorimotor circuits as a cortical basis for language. Nature Reviews Neuroscience 11, 351–360 (2010)
15. Rohde, D.L.T., Plaut, D.C.: Connectionist models of language processing. Cognitive Studies 10(1), 10–28 (2003)
16. Roy, D.K., Pentland, A.P.: Learning words from sights and sounds: A computational model. Cognitive Science 26(1), 113–146 (2002)
17. Steels, L., Spranger, M., van Trijp, R., Höfer, S., Hild, M.: Emergent action language on real robots. In: Language Grounding in Robots, ch. 13, pp. 255–276. Springer, New York (2012)
18. Wermter, S., Panchev, C., Arevian, G.: Hybrid neural plausibility networks for news agents. In: Proc. National Conference on Artificial Intelligence (AAAI 1999), Orlando, US, pp. 93–98 (July 1999)
19. Yamashita, Y., Tani, J.: Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. PLoS Computational Biology 4(11), e1000220 (2008)

Unsupervised Online Calibration of a c-VEP Brain-Computer Interface (BCI)

Martin Spüler[1], Wolfgang Rosenstiel[1], and Martin Bogdan[1,2]

[1] Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Germany
[2] Computer Engineering, University of Leipzig, Germany

Abstract. Brain-Computer Interfaces (BCIs) can be used to give paralyzed patients a means for communication. But so far, only supervised methods have been used for calibration of an online BCI. In this paper we present a method that allows to calibrate a BCI online and unsupervised. Based on offline data we show that the unsupervised calibration method works and validate the results in an online experiment with 8 subjects, who were able to control the BCI with an average accuracy of 85 %. We thereby have shown for the first time that an online unsupervised calibration of a BCI is possible and allows for successful BCI control.

Keywords: Brain-Computer interface (BCI), unsupervised learning.

1 Introduction

A Brain-Computer Interface (BCI) is a device that enables a user to control a computer by pure brain activity, which is usually recorded by electroencephalography (EEG). The main application for BCIs is to give paralyzed people a means to communicate, but so far, there are no reports for successful BCI control in complete locked-in patients [1].

Recently, we could show a BCI based on code-modulated visual evoked potentials (c-VEPs) to achieve very high communication speeds that made it possible for subjects to spell an average of 21.3 error-free letters per minute [2]. While this BCI used an unsupervised online adaptation, it still depended on a supervised calibration, for which labeled data is needed to calibrate the BCI on the users brain activity. When looking at BCIs that use other paradigms like motor imagery or P300, there are also different unsupervised adaptation methods [3], but they all depend on a supervised calibration, which needs labeled data.

So far, Eren et al. [4] are the only ones, who have shown that a BCI can be calibrated completely unsupervised without the need for labeled training data. Using Gaussian Mixture Models, they have shown in an offline analysis of motor imagery BCI data that their method works for 3 out of 6 subjects.

In this paper, we present a method for completely unsupervised calibration of a c-VEP BCI and show it to work for all our subjects in an online study. We further discuss how unsupervised calibration might be useful for complete locked-in patients, for whom supervised calibration does not work [1].

© Springer-Verlag Berlin Heidelberg 2013

2 Methods

The c-VEP BCI that we used is based on the system we described in previous publications [2,5]. It consists of 32 visual stimuli (called targets), which are modulated by a pseudo-random code with a length of 63 bit. Each target is modulated with the same code, but the code is circular-shifted by a different number of bits for each target. When the subject is looking at one of the stimuli, a c-VEP can be found in the EEG signals. Based on multiple trials, the average c-VEP waveform can be extracted. By circular-shifting the average c-VEP waveform, a template for each target can be obtained, which represents the average c-VEP waveform that is expected when the subject looks at the corresponding target. To identify which of the targets the user wants to select, the EEG-signal is compared to all templates and the template which is closest to the measured EEG signal is chosen.

While the previous system with supervised calibration used 32 targets, we only used two targets (J and W) for the unsupervised calibration. Regardless of the number of targets used for calibration, the system can be tested and used with 32 targets after calibration is finished.

To achieve an unsupervised calibration of the c-VEP BCI, the first step is to perform an unsupervised channel selection to find the EEG channel on which the c-VEP is strongest. If that channel is found, the templates can be generated in an unsupervised manner, as described below.

2.1 Unsupervised Channel Selection

The goal of the unsupervised channel selection is to find the channel for which the c-VEP is strongest, which means finding the channel for which the c-VEP has the lowest variance.

Since only two shifts are possible (20 bit and 46 bit, representing targets J and W), a new dataset is created that contains each trial of the calibration dataset twice, one shifted by -20 bit and the other by -46 bit. By doing this, the data should contain 3 clusters: one cluster containing the data that is shifted with the correct shift, one cluster for data with a true shift of 20 bit that was shifted by -46 bit, and one cluster with a true shift of 46 bit that was shifted by -20 bit. Since the cluster with the data shifted by the correct number of bits should contain twice the number of trials than each of the other 2 clusters, a one-class Support Vector Machine (OCSVM) [6] can be trained to reject the smaller clusters as outliers.

Therefore, the data for each channel is normalized to have a mean of zero and a variance of 1. The normalized data is used to train a OCSVM with a linear kernel and $\nu = 0.4$ to find a hyperplane which separates the one large cluster from the two small ones. The size of the margin can then be used as an approximation of the variance. A larger margin means a smaller variance of the c-VEP data and thereby a stronger c-VEP response. The channel for which the margin is largest is then chosen for the unsupervised template generation.

2.2 Unsupervised Template Generation

When the channel with the strongest c-VEP is found, the k-means algorithm [7] is applied on the calibration dataset using the respective channel to find two clusters that represent the data for the two targets. Since the k-means algorithm is only used to find clusters, it is unknown, which cluster represents which class. To assign classes to both clusters, two leave-one-out estimations are performed, in which both of the two possible assignments are tested (Cluster A -> Class A or Cluster A -> Class B). For each of the n folds of the leave-one-out estimation, templates are generated using the calibration data of *n-1* trials (with the labels associated with one of the possible assignments), and the remaining trial is classified by choosing the template which has the highest correlation. Due to the circular-shift property of the c-VEP BCI, templates were generated for 32 classes with each class having an additional 2 bit shift (total length of modulating code was 63 bit). For the classification of the remaining trial, also 32 possible classes were used.

Since templates for all 32 classes are generated, but only data containing 2 classes is used for the calibration, and the difference in the shift of both classes is uneven depending on the direction of the shift (46-20 = 26 bit or 20-46 = (-26 mod 63) = 37 bit), the assignment with the highest estimated accuracy is the one that assigns the correct class to each cluster.

Thereby class labels are available for each target. Although the unsupervised channel selection is an important step for the unsupervised template generation to work, we empirically found a subsequent channel selection based on the estimated labels to further improve the results. Therefore, leave-one-out estimations are performed for all channels to find the channel that yields optimal results.

After class labels and the best channel for classification are known, templates and spatial filter can be generated as explained in our previous publication [5].

2.3 Offline Analysis

To evaluate the unsupervised calibration method, data recorded from a previous c-VEP study [2] was used for an offline analysis, in which an online experiment was simulated. The first 64 trials (used for supervised calibration in the previous online study) were used for unsupervised calibration. Since the data was recorded with the subjects attending each of the 32 targets 2 times, the trials were shifted in a way that half of the trials had a shift corresponding to target letter J with a shift of 20 bit and the other half corresponding to target letter W with a shift of 46 bit.

The remaining 576 trials of the session were used to estimate the accuracy. Two different approaches were tested: One approach, in which the shift of the trials did not change and thereby all 32 classes were present in the data. For the other approach, the trials were shifted similar to the calibration data to simulate the use of a 2-class c-VEP BCI system.

In addition, the benefit of the unsupervised channel selection was evaluated by replacing the unsupervised channel selection with a fixed selection of either electrode P4 or PO3, which are the electrodes where the c-VEP is strongest on average.

Since the signal-to-noise ratio of the signal may be too low to allow for unsupervised calibration, tests were run in which multiple trials were averaged, similar to the method using multiple sequences in the popular P300 BCI speller. x subsequent trials were averaged to generate one new trial, thereby decimating the total number of trials by a factor of x. While still the first 64 trials of the new dataset were used for calibration, the number of test trials varied depending on x.

2.4 Online Experiment

To validate the results from the offline analysis, another online experiment was conducted, in which a c-VEP BCI with 2 targets was calibrated unsupervised and tested afterwards. Eight subjects (mean age 25, 2 female) participated in this experiment, with none of them having previous BCI experience.

Calibration was done in a co-adaptive manner, in which the first 64 trials were used for unsupervised calibration and the classifier was updated after every trial, so that feedback could be given also during the unsupervised calibration. The subjects were instructed to decide freely, which of the two targets to attend, but not to switch the target consistently every time and not to attend one target for more than 5 consecutive trials.

After calibration was finished, the accuracy of the calibrated c-VEP BCI was tested in another 128 trials. For testing the BCI with 2 targets, the subjects were instructed to alternate between both targets.

Calibration and testing were done two times. One time without averaging over trials and one time with averaging 2 trials.

Since the subjects could freely decide what targets to attend during the calibration, the data could not be used to simulate a supervised adaptation. Instead, another experiment was run with the same subjects, in which a supervised calibration [5] was used with 2 targets and 64 trials. To test the accuracy, the BCI was run with 2 targets for another 128 trials. This experiment was only done once without averaging for each subject.

3 Results

3.1 Offline Analysis

The accuracies obtained during a simulated online session with 2 targets after an unsupervised calibration are displayed in table 1 for a different number of trials averaged. Table 2 shows the obtained results for a simulated online session with 32 targets.

While for 2 classes, without averaging multiple trials, a mean accuracy of 90.85 % could be reached, averaging of 2 trials reached a mean accuracy of 97.4 % which is significantly better than without averaging trials ($p = 0.0051$, paired t-test). Looking at the results with a higher number of averages, the use of 3 or more averages is not significantly better than its preceding number of averages ($p > 0.1$).

For 32 classes, a mean accuracy of 76.43 % could be reached, while averaging over 2 trials yields a mean accuracy of 92.61 %, which is significantly higher ($p < 0.001$). Again the use of more than 2 averages does not yield a significant improvement ($p > 0.1$) compared to its preceding number of trials averaged.

Table 1. Offline results for unsupervised calibration with 2 targets and different number of trials used for averaging

	Number of trials averaged				
	1	2	3	4	5
AA	87.81 %	100.00 %	100.00 %	100.00 %	100.00 %
AB	65.94 %	81.25 %	69.81 %	85.00 %	85.94 %
AC	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
AD	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
AE	65.94 %	82.50 %	95.28 %	98.75 %	100.00 %
AF	96.56 %	100.00 %	100.00 %	100.00 %	100.00 %
AG	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
AH	78.75 %	90.62 %	80.19 %	77.50 %	85.94 %
AI	70.31 %	99.38 %	99.06 %	100.00 %	100.00 %
BA	97.50 %	99.38 %	100.00 %	100.00 %	100.00 %
BB	98.44 %	100.00 %	100.00 %	100.00 %	100.00 %
BC	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
BD	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
BE	95.00 %	100.00 %	100.00 %	100.00 %	100.00 %
BF	99.06 %	100.00 %	100.00 %	100.00 %	100.00 %
BG	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
BH	81.25 %	100.00 %	100.00 %	100.00 %	100.00 %
BI	98.75 %	100.00 %	100.00 %	100.00 %	100.00 %
mean	**90.85 %**	**97.40 %**	**96.91 %**	**97.85 %**	**98.44 %**

To estimate the benefit of the unsupervised channel selection, the results for the comparison of the unsupervised channel selection with channels PO3 and P4 is shown in table 3 and table 4. When pooling the results for all tested number of averages, the unsupervised channel selection in the simulated c-VEP BCI with 2 targets performs significantly better than a fixed selection of channel PO3 ($p < 0.0005$, paired t-test) and significantly better than a selection of P4 ($p < 0.05$). When simulating the use of a c-VEP BCI with 32 targets, the unsupervised channel selection still performs significantly better than PO3 ($p < 0.005$) and P4 ($p = 0.01$).

3.2 Online Experiment with 2 Targets

The results from the online experiment are shown in table 5. The results show that the BCI worked well for all subjects after unsupervised calibration with an

Table 2. Offline results for unsupervised calibration with different number of trials used for averaging. While calibration was done on data from 2 targets, data with 32 targets was used for performance evaluation.

	Number of trials averaged					supervised
	1	2	3	4	5	1
AA	75.94 %	99.38 %	100.00 %	100.00 %	100.00 %	96.88 %
AB	21.56 %	46.88 %	53.77 %	70.00 %	85.94 %	80.03 %
AC	98.44 %	99.38 %	100.00 %	100.00 %	100.00 %	98.61 %
AD	99.69 %	100.00 %	100.00 %	100.00 %	100.00 %	98.96 %
AE	18.75 %	58.13 %	87.74 %	97.50 %	100.00 %	60.24 %
AF	86.88 %	98.75 %	100.00 %	100.00 %	98.44 %	97.74 %
AG	99.69 %	100.00 %	100.00 %	100.00 %	100.00 %	99.83 %
AH	41.88 %	73.75 %	69.81 %	68.75 %	76.56 %	72.72 %
AI	43.75 %	99.38 %	99.06 %	98.75 %	100.00 %	96.18 %
BA	88.12 %	98.75 %	100.00 %	100.00 %	100.00 %	94.44 %
BB	87.19 %	96.25 %	92.45 %	100.00 %	96.88 %	97.48 %
BC	96.88 %	99.38 %	100.00 %	100.00 %	100.00 %	98.09
BD	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
BE	86.88 %	100.00 %	100.00 %	100.00 %	100.00 %	99.31 %
BF	92.81 %	98.12 %	99.06 %	100.00 %	100.00 %	94.97 %
BG	98.75 %	100.00 %	100.00 %	100.00 %	100.00 %	98.96 %
BH	55.00 %	98.75 %	100.00 %	100.00 %	100.00 %	86.98 %
BI	83.44 %	100.00 %	100.00 %	100.00 %	100.00 %	94.79 %
mean	**76.43 %**	**92.61 %**	**94.55 %**	**96.39 %**	**97.66 %**	**92.57 %**

average accuracy of 85.06 % and that averaging over 2 trials improves classification accuracy for all subjects. During the supervised calibration the subjects achieved an average accuracy of 94.43 %.

4 Discussion

In this paper, we have shown that an unsupervised calibration of a c-VEP BCI is possible and that all subjects were able to control the BCI online with an average accuracy of 85 %. Thereby it was shown for the first time that an online BCI can be calibrated in an unsupervised manner.

While the online study used a BCI with only 2 targets, we have shown in the offline analysis that a BCI with 32 targets can successfully be used after an unsupervised calibration on 2 targets. By averaging over multiple trials the accuracy can further be increased.

Although it was shown that an unsupervised calibration works well and can be used to calibrate a BCI, it does not perform better than a supervised calibration. Therefore the use of an unsupervised calibration method needs to be discussed.

So far, there are no reports of BCI working online in complete locked-in (CLIS) patients, who do not have any residual muscle control. With the transition into CLIS, the last possibility to move any muscle is lost. Thereby the patient loses

Table 3. Offline results with 2 targets and different methods for channel selection. Different number of trials were used for averaging. Unsupervised channel selection was compared to a fixed selection of channel PO3 or P4, respectively.

	Number of trials averaged				
	1	2	3	4	5
PO3	86.58 %	89.41 %	91.88 %	90.69 %	92.62 %
P4	87.81 %	92.78 %	94.24 %	94.79 %	94.53 %
unsupervised	90.85 %	97.40 %	96.91 %	97.85 %	98.44 %

Table 4. Offline results for different methods for channel selection using data with 32 targets. Different number of trials were used for averaging. Unsupervised channel selection was compared to a fixed selection of channel PO3 or P4, respectively.

	Number of trials averaged				
	1	2	3	4	5
PO3	72.87 %	81.95 %	85.43 %	85.83 %	87.24 %
P4	70.21 %	86.70 %	89.94 %	90.63 %	90.89 %
unsupervised	76.43 %	92.61 %	94.55 %	96.39 %	97.66 %

Table 5. Accuracies during the online experiment for different calibration methods: unsupervised calibration without averaging, unsupervised calibration with averaging over 2 trials, supervised calibration without averaging

	unsupervised		supervised
Subject	no averaging	average over 2 trials	
CL	64.06 %	78.13 %	95.31 %
CM	85.94 %	99.22 %	100 %
CN	78.13 %	87.50 %	92.19 %
CO	85.16 %	92.19 %	89.06 %
CP	89.84 %	95.31 %	82.81 %
CQ	93.75 %	80.47 %	100 %
CR	89.84 %	83.59 %	96.09 %
CS	93.75 %	85.94 %	100 %
mean	**85.06 %**	**87.79 %**	**94.43 %**

the last possibility to interact with the environment and he has no longer any means to follow his plans or goals. Since there is no possibility to achieve any goals, the goal-directed thinking is assumed to be extinct [1] and thereby the patient would not be able to follow any instructions that are necessary when calibrating a BCI in a supervised manner.

Unsupervised calibration might solve this problem, since it allows to calibrate a BCI and give the user feedback without the need for any goal-directed action [8]. Although it is still speculation if patients without eye-movement control can use a c-VEP BCI, it might be possible with a modified stimulus presentation (as it was already done for SSVEP BCIs [9,10]). Thereby the patient could influence his environment again and may regain the ability for goal-directed thinking, since he has now a means to communicate and achieve potential goals.

5 Conclusion

In this paper we have presented a method that allows for an unsupervised calibration of a c-VEP BCI. In an online study we have shown that all subjects were able to control the BCI with an average accuracy of 85 % after an unsupervised calibration. Although the accuracy is lower than for a supervised calibration, unsupervised methods could be used to establish communication in complete locked-in patients, for whom supervised methods does not work.

Acknowledgments. This study was partly granted by the German Federal Ministry of Education and Research (BMBF, Grant UTü 01GQ0831) and the European Union (ERC 227632-BCCI).

References

1. Kübler, A., Birbaumer, N.: Brain-computer interfaces and communication in paralysis: Extinction of goal directed thinking in completely paralysed patients? Clinical Neurophysiology 119(11), 2658–2666 (2008)
2. Spüler, M., Rosenstiel, W., Bogdan, M.: Online adaptation of a c-VEP Brain-Computer Interface (BCI) based on Error-related potentials and unsupervised learning. Plos One 7(12), e51077 (2012), doi:10.1371/journal.pone.0051077
3. Spüler, M., Rosenstiel, W., Bogdan, M.: Adaptive SVM-based classification increases performance of a MEG-based Brain-Computer Interface (BCI). In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 669–676. Springer, Heidelberg (2012)
4. Eren, S., Grosse-Wentrup, M., Buss, M.: Unsupervised classification for non-invasive brain-computer-interfaces. In: Proc. Automed Workshop, Düsseldorf, Germany, pp. 65–66 (2007)
5. Spüler, M., Rosenstiel, W., Bogdan, M.: One Class SVM and Canonical Correlation Analysis increase performance in a c-VEP based Brain-Computer Interface (BCI). In: Proceedings of 20th European Symposium on Artificial Neural Networks (ESANN 2012), Bruges, Belgium, pp. 103–108 (April 2012)
6. Schölkopf, B., Platt, C.: Estimating the support of a High-Dimensional Distribution. Neural Computation (2001)
7. Hartigan, J.A.: Clustering Algorithms, 99th edn. John Wiley & Sons, Inc., New York (1975)
8. Spüler, M., Rosenstiel, W., Bogdan, M.: Unsupervised BCI calibration as possibility for communication in CLIS patients? In: Proceedings of the Fifth International Brain-Computer Interface Meeting (2013), doi:10.3217/978-3-85125-260-6-122
9. Kelly, S.P., Lalor, E.C., Reilly, R.B., Foxe, J.J.: Visual spatial attention tracking using high-density SSVEP data for independent brain-computer communication. IEEE TNSRE 13(2), 172–177 (2005)
10. Zhang, D., Maye, A., Gao, X., Hong, B., Engel, A.K., Gao, S.: An independent brain-computer interface using covert non-spatial visual selective attention. Journal of Neural Engineering 7(1), 016010 (2010)

A Biologically Inspired Model for the Detection of External and Internal Head Motions

Stephan Tschechne, Georg Layher, and Heiko Neumann

Institute of Neural Information Processing, University of Ulm,
89069 Ulm, Germany
http://www.uni-ulm.de/in/neuroinformatik.html

Abstract. Non-verbal communication signals are to a large part conveyed by visual motion information of the user's facial components (intrinsic motion) and head (extrinsic motion). An observer perceives the visual flow as a superposition of both types of motions. However, when visual signals are used for training of classifiers for non-articulated communication signals, a decomposition is advantageous. We propose a biologically inspired model that builds upon the known functional organization of cortical motion processing at low and intermediate stages to decompose the composite motion signal. The approach extends previous models to incorporate mechanisms that represent motion gradients and direction sensitivity. The neural models operate on larger spatial scales to capture properties in flow patterns elicited by turning head movements. Center-surround mechanisms build contrast-sensitive cells and detect local facial motion. The model is probed with video samples and outputs occurrences and magnitudes of extrinsic and intrinsic motion patterns.

Keywords: Social Interaction, HCI, Motion Patterns, Cortical Processing.

1 Introduction

The recent development of computers show a clear trend towards companion properties [4,14]. Systems are demanded to improve the efficiency of human computer interaction (HCI) by adapting to the user's need, experience and moods. To achieve this, those systems must be able to interpret non-verbal communication patterns that are signalled by the user [2,5]. These patterns are to a large part contained in visual signals that can be captured by an interaction system, from which they can automatically be derived [9]. Among these visual patterns bodily and facial expressions contain important cues to emotionally relevant information, whereas both types can be either of static or dynamic nature. Optic flow has been investigated for emotion and facial analysis earlier [13,6,11,8]. However, when it comes to analysis of facial motion, extrinsic (head) and intrinsic (facial) movements are superpositioned and elicit a composite flow field of respective patterns from the observer's perspective (see Fig. 1). Subsequent classification stages profit from a segregation of these patterns. In [13], an attempt to decompose the facial optical flow into affine as well as higher-order

© Springer-Verlag Berlin Heidelberg 2013

flow patterns in order to segregate the facial motion has been proposed. In [3] head-pose and facial expressions are estimated graphics and animation. Here, an affine deformation with parallax is modelled to fit active contours using singular value decomposition. In [1] the authors propose a multi-stage landmark fitting and tracking method to derive face and head gestures.

In this paper we propose a novel mechanism to detect occurrences of basic components of motion from optic flow. The method is studied on the example of head movements and dynamic facial expressions which both cause optic flow at the observer position. We model mechanisms of signal processing in early and intermediate stages of visual cortex to provide robust automatic decomposition of extrinsic and intrinsic facial motions. We demonstrate how occurrences of extrinsic as well as intrinsic components are robustly derived from an optic flow field. Our approach contrasts others by neither requiring face detection or a previous learning phase. Additionally, processing of multiple persons comes at no extra cost.

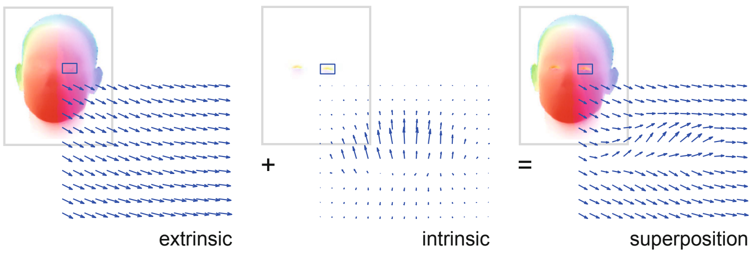


Fig. 1. Visual flow at the observer position is a superposition (*right*) of extrinsic (head, *left*) and intrinsic (facial, *middle*) motion. Subsequent processing benefits from separated processing of both sources.

2 Visual Representations of Head Movements

The instantaneous motion of a three-dimensional surface that is sensed by a stationary observer can be represented by the linear combination of its translational and rotational motion components [10], as well as non-rigid motion caused by object deformations. Any of these cause visual motion in the image plane. In this paper we focus on the analysis of facial and head motion of an agent in a communicative act by means of optic flow. This motion is composed of the *extrinsic* (head) *motion* caused by translations and rotations and the superimposed internal motion of facial components (*intrinsic motion*). We assume that any translational extrinsic motion of the head has been compensated to fixate the head in the middle of the image so that the world coordinate system is centered in the moving head. We aim at spatial processing of the resulting patterns to individually detect the extrinsic and intrinsic components. For a rotation of a simple head model around the Y-axis, an arbitrary surface point $P = (x, y, z)^T$ appears on a rotational path in space (see Fig.2) with frequency ω and periodic length T. P is uniquely defined by its radius r and vertical component y at time t, when a distance d to the observer is assumed:

$$P(t, r, y) = r \cdot \begin{pmatrix} cos\,\omega t \\ y \\ sin\,\omega t \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} \qquad \omega = \frac{2\pi}{T} \tag{1}$$

In the following we assume that $y = 0$ and $r = 1$. Application of the projection model with $x = f \cdot X/Z$ and $y = f \cdot Y/Z$, given the focal length f of the camera yields the projected coordinates P_{proj} in image plane for the observer:

$$P_{proj}(t, r) = f \cdot \begin{pmatrix} \frac{r\ cos\ \omega t}{r\ sin\ \omega t - d} \\ 0 \end{pmatrix} \tag{2}$$

If we assume constant angular speed, the apparent image speed of the projection of P is

$$\frac{\partial P_{proj}(t, r)}{\partial t} = \begin{pmatrix} \frac{rsin\omega t + d - 1}{r^2 sin^2 \omega t - 2dr\ sin\omega t + d^2} \\ 0 \end{pmatrix} \tag{3}$$

which yields a characteristic motion pattern, see Fig. 2. The apparent motion on the positive half circle, where the facial surface is oriented towards the observer, leads to a generic speed pattern. For a frontal motion from right to left the pattern is composed by an acceleration, followed by maximum frontal motion, and a symmetric deceleration patch. This pattern corresponds to the *speed gradients* as investigated by [12] and is also depicted in Fig. 2. For symmetric and bounded objects image patches of increasing and decreasing speeds are juxtaposed reflecting appearance and disappearance of the surface markings on a convex rotating body surface. In Sec. 2.1 we suggest a filtering mechanism for these arrangements of *speed gradients* which is tuned to such symmetric arrangements of image motions with symmetric speed changes.

Facial motion, on the other hand, is caused by actions of facial muscles, e.g. during verbal communication, eye blinks, or while forming facial expressions. These spatio-temporal changes occur as deformations on a smaller temporal and spatial scale compared to the size of the face and are characterized by changes in motion direction and/or speed relative to its coherent surrounding motion. In order to analyze such intrinsic facial motions, we reasoned that a center-surround mechanism for the filtering of motion patterns within the facial region will indicate occurrences of intrinsic motions, see Sec. 2.2.

2.1 A Model of Cortical Motion Gradient Detection

In the following we describe the implementation details of our model cells for detecting motion patterns that are characteristic for extrinsic motions corresponding to rotations around the X- or Y-axis, respectively, namely patterns

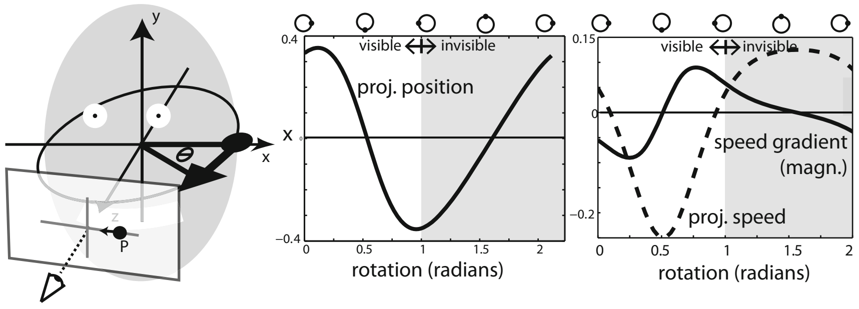


Fig. 2. *Left:* Projection model of one point on a head's surface and its trajectory in the projection when the head is rotating. *Middle:* X-position over rotation angle of point P. *Right:* One-periodical plot of speed over rotation angle (*dashed*) and speed gradient (*solid*) of a point when rotating on a circular path around Y axis. Point P is closest at position 1.0. Due to foreshortening effects, characteristic speed gradients occur where the point approaches or retreats.

containing speed gradients. All presented detectors need a visual motion field $\mathbf{u}$ which is transformed into a log-polar representation, the velocity space $\mathbb{V}$. In velocity space, motion is separably represented by speed $s_p = log\,\|\mathbf{u}\|$ and direction $\tau_p = tan^{-1}(\frac{u_y}{u_x})$. This representation allows selecting image locations containing certain motion directions, speeds, or both, which will be fundamental features for the upcoming design of gradient cells. Filters for speed ψ and direction θ with tuning widths σ are defined by

$$F_S(\mu,\nu) = exp(-\frac{(\mu-\psi)^2}{2\sigma_\psi^2}) \tag{4}$$

$$F_\theta(\mu,\nu) = exp(-\frac{(\nu-log(\theta))^2}{2\sigma_\theta^2}) \;\; \forall(\mu,\nu)\in\mathbb{V}. \tag{5}$$

Gradient cells $M_p^{+/-}$ at image position p respond when two conditions are served, see Fig. 3: First, conjunctive input configurations need to match their tunings for speed and directions, and second, an increase or decrease in speed along an axes corresponding to their directional exists. This increase or decrease is reflected in a simultaneous stimulation of two speed- and direction-tuned cells that are spatially arranged to build the desired speed gradient. The speed- and direction-tuned subcells M_p are derived from the given motion field by applying a filter F in velocity space representation. Each cell response incorporates a divise normalisation component in order to keep responses within bounds.

$$M_p = \mathbb{V}_p \cdot F \tag{6}$$

$$M_p^+ = M_{p-\Delta p}\cdot M_{p+\Delta p}\cdot(\epsilon + M_{p-\Delta p} + M_{p+\Delta p})^{-1} \qquad (0<\epsilon\ll 1) \tag{7}$$

Responses to flow gradients of opposite polarity are subsequently combined by *AND*-gating to build a curvature detector. These cells operate at spatially juxtaposed locations with component offsets along their directional preference depending on the spatial scale of the filter kernels.

$$C_p = M^{+}_{p+\Delta p} \cdot M^{-}_{p-\Delta p} \cdot (\epsilon + M^{+}_{p+\Delta p} + M^{-}_{p-\Delta p})^{-1} \tag{8}$$

In order to make the response more robust and selective to motion direction this curvature response is combined with the output of a motion integration cell M_p. The final response is thus defined by

$$R_p = M^{+}_{p+\Delta p} \cdot M_p \cdot M^{-}_{p-\Delta p} \cdot (\epsilon + M^{+}_{p+\Delta p} + M_p + M^{-}_{p-\Delta p})^{-1} \tag{9}$$

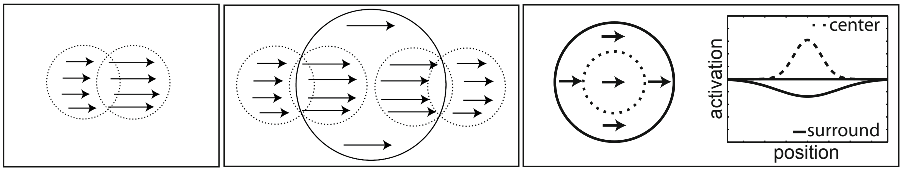


Fig. 3. *Left*: Gradient Cell Mp^{+}. *Middle:* The full model cell circuit for detecting rotational motion patterns. Oppositely tuned gradient cells are combined with cells sensitive to uniform motion patterns. *Right*: Center-surround cell for motion discontinuity detection with two centered subcells with different spatial tunings and integration weights for center and surround cell.

2.2 Model Mechanisms for Motion Contrast Detection

Local facial motions can be accounted by mechanisms that operate on a smaller scale within the facial projection into the image plane. To detect *intrinsic motion*, we propose cells that are sensitive to local changes in speed and direction. These motion patterns are produced in the facial plane while the person is talking or during other facial actions. Here, we employ a center-surround interaction mechanism of motion sensitive cells that are able to detect local variations in visual flow, but won't be sensitive to large uniform flow fields. Such a sensitivity can be generated by cells with antagonistic center-surround motion sensitivity. The input integration of velocity responses is defined by weighted kernels Ω with different spatial scale dimensions operating on responses of motion and speed selective filters. Integration over N directions yields the activation for a direction-insensitive motion contrast cell.

$$D^{sub}_{p,\theta} = \mathbb{V}_p * F_{\bar{\mu}} \tag{10}$$

$$D_{p,\theta} = \frac{max\left(0, D^{sub}_{p,\theta} * \Omega_{inner} - D^{sub}_{p,\theta} * \Omega_{outer}\right)}{\epsilon + D^{sub}_{p,\theta} * \Omega_{inner} + D^{sub}_{p,\theta} * \Omega_{outer}} \tag{11}$$

$$D_p = \sum_{\theta \in n} \|D_{p,\theta}\| \quad n = \{1, \ldots, N\} \tag{12}$$

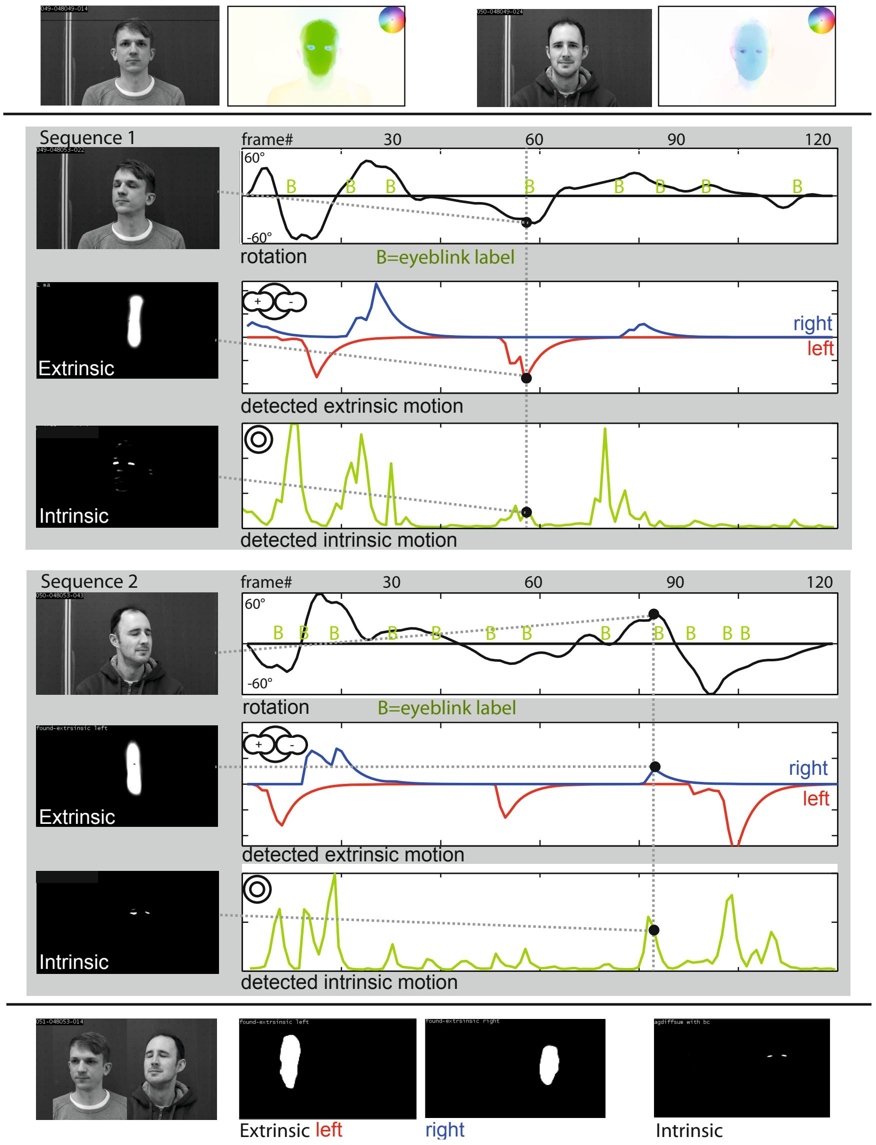


Fig. 4. Results for the processing of two interaction sequences with unconstrained head motion. *Top four images:* Sequence snapshots and color-coded optic flow fields. *Large central part*: Manually labeled head rotation and eye blinks (*1st&4th plot*), activation of rotation-selective cells (*2nd&5th plot*) and activation of cells for intrisic motion (*3rd&6th plot*). Single processing snapshots are arranged on the left, their points of reference are indicated by dotted lines. *Bottom four images*: Snapshot of synthesized video with a test sequence containing two persons to demonstrate simultaneous processing. In the selected snapshot the two persons rotate their head differently, which is reflected in the activations of respective cells. The right person additionally blinks with his eyes.

3 Results

Our model was probed with short video sequences containing extrinsic or intrinsic motion. Optic flow was generated from these scenes by using a high quality estimation method [7], see top four images of Fig. 4. The optic flow was then transformed into velocity space representation and presented to proposed model cells. The middle section of Fig. 4 shows results for sequences of unconstrained motion, where persons could move the head *ad libitum*. The persons were told to blink with their eyes randomly to simulate small intrinsic motions. Head rotation and eye blinks were manually annotated. Extrinsic motion was then detected by the proposed model cells. Plots of the activations are also shown in the figure. The rotation of the head is recovered and a differentiation is made between leftward and rightward motion. Also, detectors for intrinsic motion show activations that correlate well with eye blink labels. Responses of model cells were temporally filtered with a moving average operation to eliminate spurious temporal artefacts due to errors in the flow estimation process. Note that the detectors not only indicate the *presence* of an occurrence but also the *position*. This also holds when a sequence with multiple persons is processed, see bottom four images of Fig. 4.

4 Conclusion and Discussion

We propose a biologically inspired model of motion processing that is sensitive to occurrences of motion patterns specifically produced by extrinsic or intrinsic head motions in user interaction scenarios. These movements are perceived as composite motion signals by the observer. Correct interpretation regarding user dispositions and non-verbal communication patterns from visual signals requires segregated processing of both sources. We propose networks of cortical motion processing to detect the individual occurrences of intrinsic and extrinsic motion. The model incorporates a stage of detectors for rotations around $X-$ and $Y-$ axes that include sensitivity to motion gradients and motion direction. Center-surround mechanisms indicate intrinsic motions by detecting local contrasts that are produced by local motions of facial components. We probed the model with realistic input sequences containing head rotations and eye blinks to represent both motion components. Both proposed detectors work well on facial images and segregate composite facial motion into their extrisinic and intrinsic components. Our approach does not need training or face detection and can process multiple persons in a sequence simultaneously. Subsequent processing steps profit from the detection of occurrence and position of movement components. In contrast to other approaches our proposed model is independent from highly specialised models, tracking, learning from examples or large optimisation stages to derive the presented results. By this we demonstrate how human-computer-interfaces profit from biologically inspired mechanisms of motion processing by detecting occurrences of specific motion patterns from an optic flow field. Future work will include a validation of the approach for more generic shape-from-motion tasks.

This work has been supported by grants from the Transregional Collaborative Research Center SFB/TRR62 'A Companion Technology for Cognitive Technical Systems' funded by the German Research Foundation (DFG).

References

1. Akakin, H.C., Sankur, B.: Robust Classification of Face and Head Gestures in Video. Image and Vision Computing 29, 470–483 (2011)
2. Argyle, M.: Bodily Communication, vol. 2. Routledge, London (1988)
3. Bascle, B., Blake, A.: Separability of Pose and Expression in Facial Tracking and Animation. In: Proc. Int. Conf. on Computer Vision, pp. 323–328 (1998)
4. Benyon, D.: Designing Interactive Systems, 2nd edn. Pearson, Addison–Wesley, London (2010)
5. Benyon, D., Mival, O.: Landscaping Personification Technologies: From Interactions to Relationships. In: CHI Ext. Abs. on Human Factors in Computing Systems, pp. 3657–3662 (2008)
6. Black, M., Yacoob, Y.: Recognizing Facial Expressions in Image Sequences Using Local Parameterized Models of Image Motion. International Journal of Computer Vision 25(1), 23–48 (1997)
7. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High Accuracy Optical Flow Estimation Based on a Theory for Warping. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3024, pp. 25–36. Springer, Heidelberg (2004)
8. Donato, G., Bartlett, M.S., Hager, J.C., Ekman, P., Sejnowski, T.J.: Classifying Facial Actions. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(10), 974–989 (1999)
9. Frith, C.D., Wolpert, D.M.: The Neuroscience of Social Interaction: Decoding, imitating, and influencing the actions of others. Oxford University Press (2004)
10. Longuet-Higgins, H.C.: A Computer Algorithm for Reconstructing a Scene from Two Projections. Nature 293, 133–135 (1981)
11. Niese, R., Al-Hamadi, A., Farag, A., Neumann, H., Michaelis, B.: Facial Expression Recognition Based on Geometric and Optical Flow Features in Colour Image Sequences. IET Computer Vision, 1–11 (2011)
12. Orban, G.: Higher Order Visual Processing in Macaque Extrastriate Cortex. Physiology Review 88(88), 59–89 (2008)
13. Rosenblum, M., Yacoob, Y., Davis, L.S.: Human Expression Recognition from Motion Using a Radial Basis Function Network Architecture. IEEE Transactions on Neural Networks 7(5), 1121–1138 (1996)
14. Wendemuth, A., Biundo, S.: A Companion Technology for Cognitive Technical Systems. In: Esposito, A., Esposito, A.M., Vinciarelli, A., Hoffmann, R., Müller, V.C. (eds.) COST 2102. LNCS, vol. 7403, pp. 89–103. Springer, Heidelberg (2012)

Cortically Inspired Sensor Fusion Network for Mobile Robot Heading Estimation

Cristian Axenie and Jörg Conradt

Fachgebiet Neurowissenschaftliche Systemtheorie, Fakultät für Elektro- und Informationstechnik, Technische Universität München, 80290 München, Germany
{cristian.axenie,conradt}@tum.de

Abstract. All physical systems must reliably extract information from their noisily and partially observable environment, such as distances to objects. Biology has developed reliable mechanisms to combine multi-modal sensory information into a coherent belief about the underlying environment that caused the percept; a process called sensor fusion. Autonomous technical systems (such as mobile robots) employ compute-intense algorithms for sensor fusion, which hardly work in real-time; yet their results in complex unprepared environments are typically inferior to human performance. Despite the little we know about cortical computing principles for sensor fusion, an obvious difference between biological and technical information processing lies in the way information flows: computer algorithms are typically designed as feed-forward filter-banks, whereas in Cortex we see vastly recurrent connected networks with intertwined information processing, storage, and exchange. In this paper we model such information processing as distributed graphical network, in which independent neural computing nodes obtain and represent sensory information, while processing and exchanging exclusively local data. Given various external sensory stimuli, the network relaxes into the best possible explanation of the underlying cause, subject to the inferred reliability of sensor signals. We implement a simple test-case scenario with a 4 dimensional sensor fusion task on an autonomous mobile robot and demonstrate its performance. We expect to be able to expand this sensor fusion principle to vastly more complex tasks.

Keywords: Cortical inspired sensor fusion, graphical network, local processing, mobile robotics.

1 Introduction

Environmental perception enables a physical system to acquire and build an internal representation of significant information within its environment. As an example of such an internal state, accurate self-motion perception is an essential component for spatial orientation, navigation and motor planning for both real and artificial systems. A system can build its spatial knowledge using a combination of multiple sources of information, conveyed from self-motion related signals (e.g. odometry or vestibular signals), but also from static external environmental cues (e.g. visual or auditory) [1].

© Springer-Verlag Berlin Heidelberg 2013

In this paper we focus on one component of self-motion: heading estimation. The key aspect in this problem is to minimize interference or conflicts between multiple dynamic and static sources of perceived spatial information.

1.1 State-of-the-Art in Sensor Fusion Algorithms

Sensor fusion finds wide application in many areas of robotics, starting from object recognition to environmental mapping and localization. Many of the state-of-the-art sensor fusion methods are based on a probabilistic framework, typically using Bayes' rule to optimally combine information [2-4]. This rule defines the way to compute the posterior probability distribution of a state/quantity of interest from prior knowledge about the state and a likelihood function (e.g. knowledge about how likely different values of the state will give rise to observed sensory data). Incoming sensory data will progressively push the belief "away" from the initial prior towards beliefs (posteriors) that better reflect the data. Different methods translate the posterior into a quantity [4-5], but need to balance the trade-off between algorithmic complexity and available computing resources.

1.2 Brains Aim at Flexibility and Robustness versus Optimality

Engineering approaches for sensor fusion typically aim for optimal solutions, as demonstrated in various amazing robotic demonstrators [6]. Handling dynamic and complex features in real-world scenarios, however, engineered approaches typically require intense modifications (in the form of hand tuning), and often lack the ability to easily translate to new settings. With respect to flexibility and robustness neurobiology demonstrates vastly superior performance over today's engineered systems.

Building on principles known from information processing in the brain our proposed model for sensor fusion migrates away from classical computing paradigms: tightly coupled processing, storage and representation is replaced with a loosely interconnected network of distributed computing units, each with local memory, processing and interpretation abilities. Mutual influence among the different units that represent sensor data implicitly performs the integration of multiple sensory information.

In this paper Section 2 provides a broad description of the proposed neural network model. Following, Section 3 presents results from a simple real-world mobile robotic scenario (heading estimation), and section 4 highlights the main features of the developed model and presents ideas to extend the developed architecture.

2 Model Description

Large-scale cortical networks provide a framework to integrate evidence from neuro-anatomical and neuro-physiological studies on distributed information processing in the cerebral cortex. Elementary sensory processing functions are localized in discrete recurrently interacting cortical areas, [7-8], whereas complex functions (e.g. cue integration) are processed in parallel involving large parts of the brain [9].

Building up on such cortical architectural details we design a network model capable of finding a consistent representation of self-motion using a distributed processing scheme. Each unit in the network encodes a certain feature of self-motion, while the connectivity determines mutual influence between the units, so that information in each unit is consistent with each other. For the given task of heading estimation different sensors encode the movement in their own coordinate system and units. To extract a global motion estimate the network needs to learn correlations to maintain all information in agreement.

2.1 Generic Network Architecture

The proposed model is based on a recurrent network with dynamics given by information exchange between its nodes. Each node in the network internally stores a representation of some perceived aspect of self-motion, which is a simplified form of a more generic multi-dimensional map based representation and processing paradigm successfully applied to fast visual scene interpretation in [10]. In this work a network with multiple maps encoding various aspects of visual interpretations (e.g. light intensity, optic flow, and camera rotation) found global mutual consistency, by each map independently trying to be consistent with information in neighboring maps. Such map based representation and processing paradigms are supported by results from neuro-physiology and computational neuroscience, in terms of spatio-temporal correlations used to learn topographic maps for invariant object recognition [8], [11].

Following these principles our proposed 4D sensor fusion network is composed of four fully connected nodes, which mutually influence each other. Information stored in each node is represented by a neural population code [13], in which each neuron represents a preferred angular value. Nodes' mutual influence is characterized by the physical / mathematical relationship between the four sensor representations. Relationships between nodes can be interpreted as modulating neuronal connectivity between the network's different populations [13]. To minimize mismatch between values encoded in the nodes, the network will push all representations towards an equilibrium state in which simultaneously all relationships are - approximately - satisfied. This resulting state shows the inferred quantities in each of the nodes' representation with respect to both sensor connection and internal network belief.

2.2 Dynamics

Network dynamics is described by a random gradient descent update process at every convergence step, so that the value in a selected node will take a small step towards minimizing the mismatch with a relationship in which the node is involved. Each node represents a single value in form of a neuronal population code. Relations between the nodes in the network are arbitrary functions of v variables, $f_{mi,mj,\dots,mv}(t) = 0$, where $v<N$ (the number of nodes). The generic update rule for a node m_i with respect to the relationship with node m_j in a network with N nodes, given E (the mismatch between node m_i and its relationship with m_j) and the update rate $\eta(t)$ is:

$$m_i(t+1) = m_i(t) - \eta(t) \cdot E_{m_i,m_j}(t), \quad E_{m_i,m_j}(t) = m_i(t) - f_{m_i,m_j}(t) \tag{1}$$

The convergence speed is determined by the update rate *η(t)*. In every iteration the update rate takes into account the external information from the sensor or the other node respectively, and the network's internal belief, to modulate the influence of that sensor or node:

$$\eta_{i,j}(t+1) = \eta_{i,j}(0) \cdot \frac{\sum_{k=1}^{N\text{-}\{i\}} E_{m_i,m_k}(t)}{N \cdot E_{m_i,m_j}(t)} \tag{2}$$

Eq. 2 shows that the update rate *η(t)* adjusts itself proportionally to the mismatch from the network's belief (numerator) and penalizes the influence from a sensor or node if the values it provides conflicts with the network's belief.

2.3 Sensor Fusion Network

For our heading estimation scenario the four nodes in the network represent four sources of information providing heading measurements. We consider inertial heading (from gyroscope yaw axis, "SG"), magnetic heading (from compass, "SC"), odometry heading (computed from wheel encoder information, "SW") and vision heading (from an on-board camera tracking a distal marker on the ceiling, "ST"); see Fig. 1, gray outside boxes and Fig. 2 robot setup. Each of the sensors provides raw data, which is preprocessed to align to a common unit (e.g. by integration, shift/offset). After the preprocessing stage all sensors are connected to their respective network node and data flows into the network. Each node in the network (G, C, T, W) maintains an independent heading estimate. The network computes a global heading estimate by balancing its internal belief and new sensor contributions. Every convergence step each node's information is updated as given in eq. 1. For example, the gyroscope representation update rules for the given network topology are:

$$G(t+1) = (1 - \eta_{G,C}(t)) \cdot G(t) + \eta_{G,C}(t) \cdot C(t), \tag{3}$$

$$G(t+1) = (1 - \eta_{G,W}(t)) \cdot G(t) + \eta_{G,W}(t) \cdot W(t), \tag{4}$$

$$G(t+1) = (1 - \eta_{G,T}(t)) \cdot G(t) + \eta_{G,T}(t) \cdot T(t), \tag{5}$$

$$G(t+1) = (1 - \eta_{G,SG}(t)) \cdot G(t) + \eta_{G,SG}(t) \cdot SG(t), \tag{6}$$

The network architecture used in our scenario is depicted in Fig. 1.

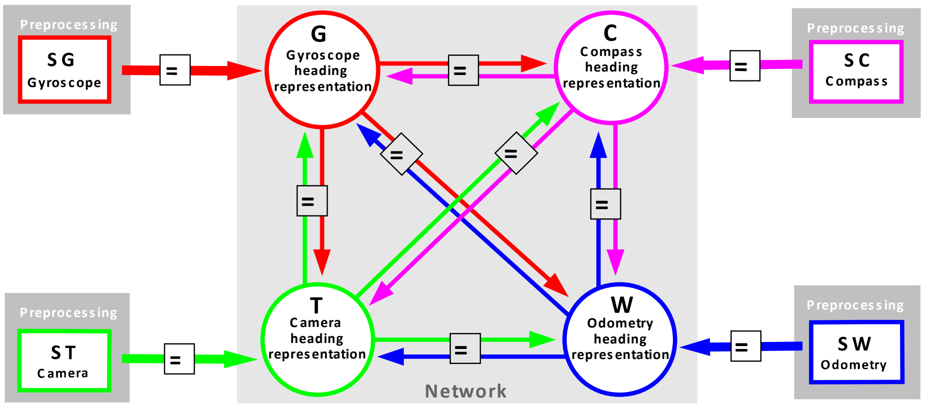

Preprocessing
SG
Gyroscope
G
Gyroscope heading representation
C
Compass heading representation
Preprocessing
SC
Compass
=
Preprocessing
ST
Camera
T
Camera heading representation
W
Odometry heading representation
Preprocessing
SW
Odometry
Network

Fig. 1. The network architecture used in the 4 dimensional sensor fusion scenario

One can easily observe the recurrent flow of information within the network and from sensors into the network. The chosen structure allows balancing contributions from each sensor within the network, resulting in a global consistent representation.

2.4 Comparison with State-of-the-Art

Relating our method with state-of-the-art probabilistic sensory fusion mechanisms we we address three key aspects: complexity, flexibility and robustness (Table 1).

Table 1. Comparison between state-of-the-art and our proposed model for sensor fusion

Criteria	State-of-the-art approaches (Bayesian)	Proposed model
Complexity computational costs	large number of probabilities to apply probabilistic inference [2].	compute multiple simple update rules (similar to eq. 1).
Flexibility possibility to add further sensory modalities	requires parameters adjustments for additional sensory modalities; adding sensors improves performance but increases complexity [3].	sensor addition (adding more update rules/constraints) is straightforward and without complexity increase.
Robustness handling sensor failures, conflicts, and uncertainty	dedicated means to detect failures, not generally applicable; challenges in assigning probabilities in an uncertain context [2].	abnormal sensor activity can be detected and penalized by adapting η (e.g. the influence of that sensor in the global estimate).

3 Model Evaluation

We designed a test case scenario to evaluate the proposed model. This simple scenario serves as proof-of-concept; we expect to extend the system within the same framework to other (more complex!) scenarios. A customized omnidirectional mobile robot (Fig. 2) equipped with inertial measurement unit (IMU, gyroscope and compass), wheel encoders, and a vision sensor provides a minimalistic system for heading estimation.

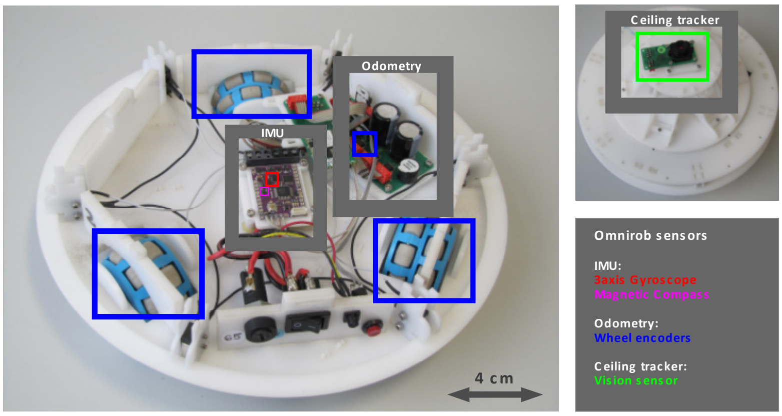


Fig. 2. Test setup for the 4 dimensional sensor fusion network: omnidirectional mobile robot

Each sensor alone is unable to provide precise heading angle measurements, as they all are affected by noise and systematic errors. The main challenge is computing a global estimate of heading direction given all unreliable sources of information. The test scenario is depicted in Fig. 3, left: the robot follows a simple predefined rectangular trajectory from start to target (top down view). Both, raw sensor data and inferred network data, are shown in Fig. 3, right.

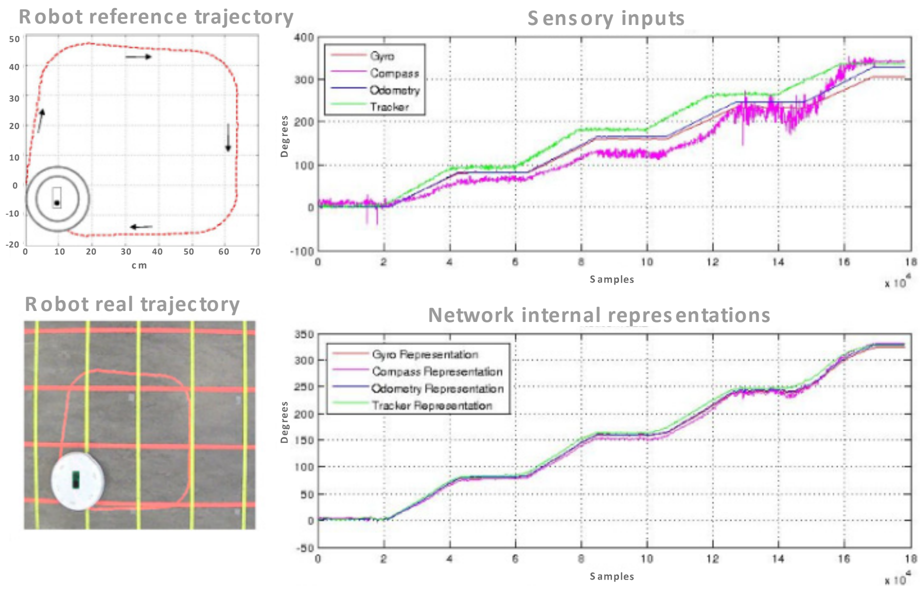


Fig. 3. Robot trajectory, raw sensors data and inferred network representations

During operation the sensors' raw data was preprocessed to align the sensors' coordinate systems (square boxes in Fig. 1). As shown in the upper right diagram of

Fig. 3 data from different sensors describe the robot's heading, but each sensor data shows a different error. The preprocessed sensor data is fed into the network, which tries to balance each sensor's contribution, as we see in the lower right diagram in Fig. 3. The network "pulls" the represented values of the nodes towards a common heading, thereby compensating for drifts and inaccuracies in individual sensor readings. The update rate is adapted (refer to eq. 2), such that "plausible" values are contributing stronger to the global estimate, while "implausible" values are penalized. Fig. 4 displays the adaptation process of the update rate in detail: to allow relaxation of the network, each sensor input data was presented for 100 network iterations. Each node in the network is fully connected to all other nodes and to an individual sensor.

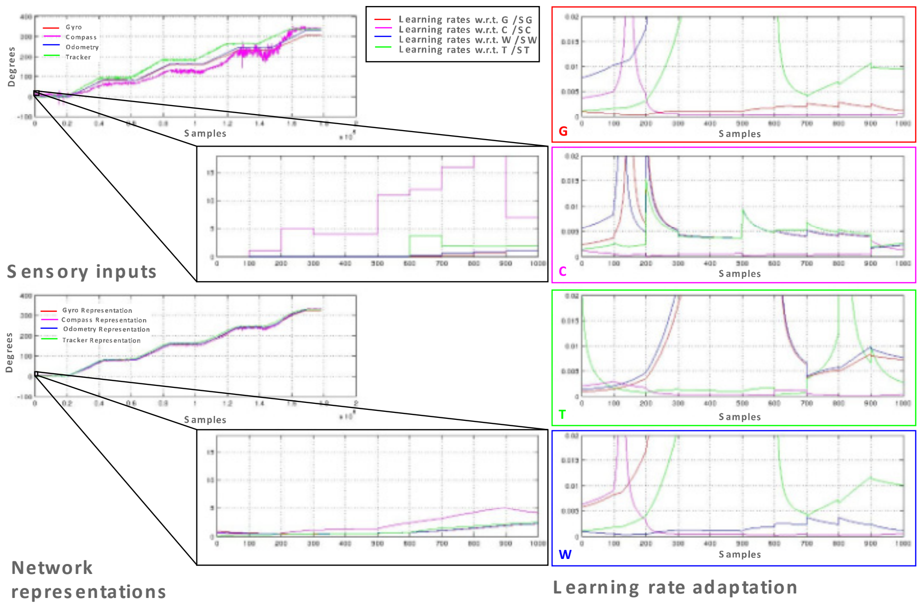


Fig. 4. Network analysis: inputs, inferred representations and learning rate adaptation

The network is able to infer which sources of information to trust by considering the mismatch (Fig. 4, right side, panels G, C, T, and W) between each local representation in a node and current representation in other nodes or its sensor. The network continuously re-evaluates and balances contributions from each sensor, so that the different representations in the network stay consistent with each other. As there is no global ground truth data source, and each of the sensors might be affected by noise and systematic errors, the network attempts to settle in a solution in which each relationship in the network is satisfied as good as possible.

4 Conclusions and Future Work

By taking inspiration from cortical distributed processing principles the presented implementation for real-world sensor fusion shows to be an alternative to state-of-the-art methods for sensor fusion, with advantages in terms of complexity, flexibility and robustness. To improve the network architecture we are currently investigating temporal relationships between different nodes in the network, allowing to remove a preprocessing step and instead feeding raw sensor data directly into the network. Such preprocessing (e.g. integration, differentiation of sensor signals) will then happen inside the network as dual relations between two representations: one node representing raw data and a further node representing the derived quantity. We will extend the network to support sensors of different types (such as pure visual input in relation to ego motion as shown here), which requires more complex relations and the representation of multidimensional data. Finally, we envision learning the network topology and thereby the underlying relations between represented quantities based on consistent real-world sensory input obtained from mobile robot exploration tasks.

Acknowledgments. The authors would like to thank Matthew Cook of INI, ETH/University Zürich for intense and fruitful discussions about map based processing and networks of relations.

References

1. Arleo, A., Rondi-Reig, L.: Multimodal sensory integration and concurrent navigation strategies for spatial cognition in real and artificial organisms. J. Integrative Neuroscience 6(3), 327–366 (2007)
2. Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics. Springer, Berlin (2008)
3. Hall, D.L., Llinas, J.: An Introduction to Multisensor Data Fusion. Proc. of the IEEE 85(1), 6–23 (1997)
4. Griffiths, T.L., Yuille, A.L.: A primer on probabilistic inference. Trends in Cognitive Sciences 10(7) (2006)
5. Körding, K.P., et al.: Causal Inference in Multisensory Perception. PLoS ONE (2007)
6. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
7. Bressler, S.L.: Understanding Cognition Through Large-Scale Cortical Networks. Current Directions in Psychological Science 11(2), 58 (2002)
8. Swindale, N.V.: How different Feature Spaces may be Represented in Cortical Maps. Network: Computation in Neural Systems 15 (2005)
9. Sporns, O.: Networks of the Brain. MIT Press (2011)
10. Cook, M., Gugelmann, L., Jug, F., Krautz, C., Steger, A.: Interacting maps for fast visual interpretation. In: Proc. of International Joint Conference on Neural Networks, pp. 770–776 (2011)
11. Michler, F., Eckhorn, R., et al.: Using Spatiotemporal Correlations to Learn Topographic Maps for Invariant Object Recognition. J. of Neurophysiology 102, 955–964 (2009)
12. Carreira-Perpinan, M.A., Lister, R.J., et al.: A Computational Model for the Development of Multiple Maps in Primary Visual Cortex. J. Cereb. Cortex 15, 1222–1233 (2005)
13. Averbeck, B.B., Latham, P.E., Pouget, A.: Neural correlations, population coding and computation. Nature Review Neuroscience (7), 358–366 (2006)

Learning Sensorimotor Transformations with Dynamic Neural Fields

Yulia Sandamirskaya[1] and Jörg Conradt[2]

[1] Institut für Neuroinformatik, RUB, Universitätstr. 150, Bochum, Germany
yulia.sandamirskaya@ini.rub.de
[2] Inst. of Automatic Control Engineering, TUM, Karlstr. 45, Munich, Germany
conradt@tum.de

Abstract. The sensorimotor maps link the perceived states to actions, required to achieve the goals of a behaving agent. These mappings depend on the physics of the body of the agent, as well as the dynamics and geometry of the environment, in which the behavior unfolds. Autonomous acquisition and updating of the mappings is crucial for robust behavior in a changing real-world environment. Autonomy of many architectures, which implement the learning and adaptation of sensorimotor maps, is limited. Here, we present a neural-dynamic architecture that enables autonomous learning of the sensorimotor transformation, involved in looking behavior. The architecture is built using Dynamic Neural Fields and is implemented on a robotic agent that consists of an eDVS sensor on a pan-tilt unit.

1 Introduction

Behavior of a biological or artificial cognitive agent may be understood in terms of the sensorimotor transformations, which map the perceived states of the environment and the agent's body onto actions, leading to accomplishment of the agent's goals. These transformations, or mappings, may be segregated into more or less independent modules based on the available sensory and motor modalities, which can be organized according to different goals, pursued by the agent [2]. Critically, the mappings between the sensory states and the actions change constantly, because of the changes in the agent's body, as in the case of a developing and aging human, or changes in the environment. Therefore, learning and adaptation of the sensorimotor maps is essential for flexible and robust generation of purposeful behavior in a real-world environment [9].

A possible mechanism to learn and update a mapping was introduces by Kohonen early on in his work on Self-Organizing Maps (SOMs) [7]. Using the mathematical mechanism of SOMs, several architectures have been introduced, which enable learning of sensorimotor mappings, involved in modeling forward and inverse models in robotic control [12,6,4,17]. Other architectures for adaptive control based on learning sensorimotor mappings use learning in multilayer neural-networks [13], incremental memory-trace update on the map based on experience [10], or error-driven learning rules (for a classical example, see [8]).

© Springer-Verlag Berlin Heidelberg 2013

This and much more work on adaptive robotic control emphasize the importance and feasibility of learning and adaptation of the sensorimotor mappings. However, all these methods share a subtle, but critical limitation, which is hindering their application outside restricted scenarios. This limitation is the lack of autonomy. For instance, in training a SOM or a neural network in an adaptive controller, robotic actions are generated by sending random commands and observing sensory states when each action is finished. Both the command and the sensory state are stored in a data vector, which is used – in most cases offline – to drive the self-organization algorithm. The autonomy of the learning process is limited here, because the mechanisms of autonomous selection, initiation, monitoring, and termination of the actions are not included in the models. The moments in time, when it is appropriate to update the map are not detected autonomously from the sensory flow. These problems of coupling of the learning processes to the perceptual and motor systems have to be solved in order to enable learning along with behavior in a real-world robotic scenario.

Autonomy of cognitive processes and their development is central in the dynamical systems approach to modeling human cognition [16]. Dynamic Field Theory (DFT) is a particular flavor of the dynamical systems approach, which has been successful in application of the cognitive models to control of robotic behavior [14,11,19]. The core element in this framework are the Dynamic Neural Fields (DNFs) – activation functions defined over topological spaces, which characterize the state of the behaving agent and its representation of the environment. Localized activity peaks emerge as stable solutions of the dynamics of DNFs and represent salient characteristics of the perceived states, as well as the goals of the upcoming motor actions.

Here, we demonstrate how the framework of DFT can be applied to learning the sensory-motor transformations involved in looking behavior. We explore how autonomous learning may be enabled in this framework along with autonomous perception and action generation. The actions are initiated and terminated autonomously based on emerging representations of intentional states. The learning process is triggered autonomously when a match between the intended and the actual sensory state is perceived and its representation is stabilized in the condition-of-satisfaction neural-dynamics. We present here an implementation of the learning architecture in a robotic system using a pan-tilt camera unit.

2 Methods: Mathematical Framework and the Dynamical Architecture

2.1 Dynamic Neural Fields

The dynamics of populations of biological neurons can be described by a continuous differential equation, which abstracts away the discreteness and the spiking nature of individual neurons, Eq. (1) [18,5,1]. Moreover, this equation can be formulated not in the space of the network of physical neurons but, instead, in the functional space of behavioral parameters, to which the neurons respond according to their tuning curves. In this formulation, an architecture of coupled dynamic neural fields is still related to neural activity in real brains, but expresses the dynamics of a neural system in functional, behavioral terms:

$$\tau\dot{u}(x,t) = -\,u(x,t) + h + \int f(u(x',t))\omega(|x'-x|)dx' + I(x,t). \qquad (1)$$

In Eq. (1), $u(x,t)$ is the activation of a dynamic neural field (DNF) at time t; x is one or several behavioral parameters (e.g., color, pitch, space, or velocity), over which the DNF is spanned; τ is the relaxation time-constant of the dynamics; h is the negative resting level, which defines the activation threshold of the field; $f(\cdot)$ is the sigmoidal non-linearity shaping the output of the neural field when it is connected to other fields or self-connected; the latter connections are shaped by the "Mexican hat" lateral interaction kernel, $\omega(|x'-x|)$, with a shot-range excitation and a long-range inhibition; $I(x,t)$ is the external input to the DNF from the sensory systems or other DNFs.

The dynamics of a DNF (Eq. (1)) has an attractor, determined by the external input, $I(x,t)$, the resting level of the field, h, and the strength of lateral interactions, specified by the kernel, $\omega(|x-x'|)$. A distinctive type the attractor of a DNF is a localized activity peak, which may be "pulled up" by the lateral interactions from a distributed input with inhomogeneities. Such peaks of activation are units of representation in Dynamic Field Theory [15]. Because of the stability and attractor properties of the DNF dynamics, cognitive models formulated in DFT may be coupled to real robotic motors and sensors and were shown to generate cognitive behavior in autonomous robots [14].

Intentionality in DFT. In order to enable autonomous activation and deactivation of dynamical attractor states in DNF architectures, each behaviorally relevant component consists of two dynamic neural fields: an intention and a condition-of-satisfaction DNF. The intention DNF is coupled to motor systems of the agent and drives its behavior by setting attractors in the low-level motor dynamics. The condition-of-satisfaction DNF receives a sub threshold input from the intention DNF and is activated by the sensory input, which matches the expected final state of the intended action. An active CoS field inhibits the intention DNF and therewith terminates the current behavior. After a brief transition instability, in which the CoS field looses its activation, the next action is selected driven by the external (bottom up) or internal (top-down) input to the intention DNF [11].

Learning in DFT. The basic learning mechanism in the DFT is the formation of memory traces of positive activation of a DNF. The memory trace is coupled back to the DNF and facilitates its activation at previously activated locations. Two DNFs may be coupled through a higher-dimensional memory structure, similar to a weight matrix in the standard neural networks. In DFT, such weight matrix is adapted through the mechanism of memory trace formation: similar to the Hebbian learning process, the coupling is strengthen between locations in two

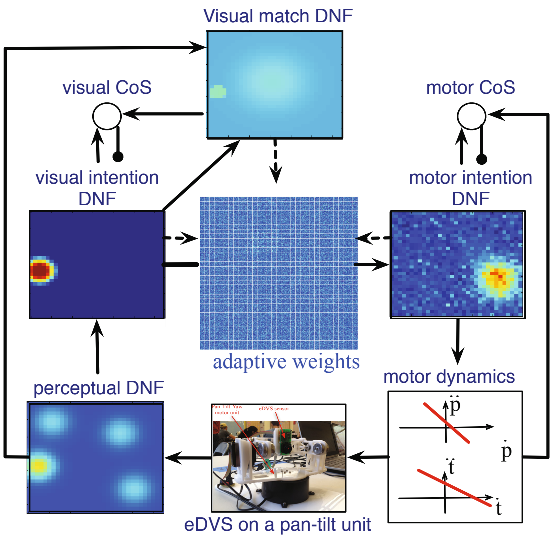


Fig. 1. The DNF learning architecture: the eDVS provides a visual input to the perceptual Dynamic Neural Field (DNF), which in its turn drives the visual intention DNF, and, through an adaptive mapping, the motor intention DNF. Visual and motor condition-of-satisfaction (CoS) nodes control the action-perception flow, and the visual match DNF detects moments, when the mapping should be updated.

DNFs, which are activated simultaneously. The learning process is functionally robust if the coupling is updated only when the behaviorally relevant states are active. In the looking architecture, presented next, we combine the elements of intentionality with learning dynamics to demonstrate autonomy of learning processes in DFT.

2.2 The DFT Closed-Loop Looking Architecture

Fig. 1 shows the DNF architecture, which both generates the autonomous looking behavior of the pan-tilt camera system and enables adaptation of the sensory-motor mapping to produce correct motor commands that move the camera toward visual targets. The architecture consists of the following dynamical structural modules.

Visual System. In the architecture, an embedded dynamic vision sensor (eDVS) [3] asynchronously generates events, which represent those pixels in the current field of view, for which the observed temporal contrast changes, e.g. because of moving objects in an otherwise static scene. Such events, generated by the hardware, provide positive input to a perceptual DNF (pDNF), in which peaks of suprathreshold activation are built at those locations where salient moving pixels are concentrated. The pDNF is input-driven, i.e. activity peaks decay if input ceases and are not sustained, new moving input induces new peak(s).

The pDNF provides input to the visual intention DNF (viDNF), in which self-sustained activity peaks may be formed. A peak in this field represents the target for the next saccade and has to be sustained for the time of the saccade, although the object representation moves in the visual field because of the camera motion. The viDNF is inhibited by the visual CoS, which signals that the saccadic movement is successfully accomplished. The viDNF is also inhibited by the motor CoS to a weaker extent, so that a new peak may be built in this field after an unsuccessful saccade, which failed to center the target.

Motor System. A peak of positive activation in the viDNF induces an activity peak in the motor intention DNF (miDNF) through a matrix of adaptive weights, which map locations in the viDNF to locations in the miDNF. The learning mechanism, active in this coupling structure will be described in the section on Sensorimotor transformation.

Activity peaks in the miDNF set attractors for the motor dynamics of the looking behavior according to Eq. (2):

$$\tau p\ddot{a}n(t) = -\,p\dot{a}n(t) + \xi_{pan}(t), \quad \tau t\ddot{i}lt(t) = -t\dot{i}lt(t) + \xi_{tilt}(t), \tag{2}$$

where $\xi_{pan}(t)$ and $\xi_{tilt}(t)$ are attractors for the rate of change of the pan and the tilt of the camera head unit, set according to Eq. (3):

$$\begin{aligned} \xi_{pan}(t) &= c_1 \iint k f(u_{mot}(k,l,t))dkdl, \\ \xi_{tilt}(t) &= c_2 \iint l f(u_{mot}(k,l,t))dkdl. \end{aligned} \tag{3}$$

Here, k and l are the two dimensions of the miDNF, which correspond to the pan and tilt velocities, respectively. The ξ_{pan} and ξ_{tilt} are estimations of the location of the activity peak in the miDNF along its two dimensions; c_1 and c_2 are scaling constants.

A peak in the miDNF sets a non-zero attractor for the pan and tilt velocities. As long as the velocity variables approach this attractor, the camera moves. When the attractor is reached, the motor CoS node, Eq. (4), is activated and inhibits the miDNF. When activity in the miDNF ceases, the motor attractors are set to zero (according to Eq. (3)).

$$\tau \dot{v}_{cos}(t) = -v_{cos}(t) + h_{cos} + c_{exc} f(v_{cos}(t)) + c_m f_{\iint}(u_{mot}) + c_a f_{diff}. \tag{4}$$

In Eq. 4, $v_{cos}(t)$ is activation of the motor CoS node for either pan or tilt movement; $f_{\iint}(u_{mot}) = \iint f(u_{mot}(k,l,t))dkdl$ is the peak-detector for the miDNF; $f_{diff} = f(0.5 - |\xi_{pan} - p\dot{a}n|)$ is a detector, activated when the state variable for the pan or the tilt dynamics reaches the respective attractor; c_m and c_a are scaling constants for these two contributions, c_{exc} is the strength of self-excitation of the motor CoS node.

Following the dynamics of Eq. (2-4), the "saccades" are produced with different horizontal and vertical amplitudes depending on the location of the activity peak in the miDNF.

Sensorimotor Transformation. Initially, the coupling between the viDNF and the miDNF is modeled by a random connectivity matrix. The coupling structure is updated directly after a successful saccade, when the (still active) location in the visual intention DNF and the (still active) location in the motor intention DNF correspond to a correct mapping. The strength of the memory-trace activation in the respective location in the coupling structure is updated according to a simple Hebbian-like learning rule ("fire together – wire together"), gated by the activity in the visual match DNF (vmDNF), Eq. (5).

$$\begin{aligned}\tau_l \dot{T}(x,y,k,l) &= \lambda \int f(u_{match}(x,y))dxdy \cdot \\ &\cdot \Big(-T(x,y,k,l) + f(u_{vis}(x,y)) \times f(u_{mot}(k,l)) \Big) \end{aligned} \qquad (5)$$

The coupling structure $T(x,y,k,l)$ (time-dependence is omitted in the equation) between the viDNF, $u_{vis}(x,y)$, defined over image coordinates (x,y), and the miDNF, $u_{mot}(k,l)$, defined over motor coordinates, k (pan) and l (tilt), retains its values if the vmDNF, $u_{match}(x,y)$, is salient. If there is a positive activation in the vmDNF (i.e., the visual input from the target landed in the central part of the pDNF, see Fig. 1), the integral before the learning term shunts the change in the mapping to be non-zero. The learning equation sets an attractor for $T(x,y,k,l)$ at the values of positive correlation between the two intention DNFs, calculated as a sum between the output of the viDNF, expanded along the dimensions of the miDNF, and the output of the miDNF, expanded in the dimensions of the viDNF, augmented with a sigmoidal threshold function (this neural-dynamic operation is denoted by the $\times$ symbol in Eq. (5)).

3 The Learning Experiments

Fig. 2 (left) shows an exemplary time-course of the pan component of several saccadic movements. The upper plot shows the time-course of the pan-velocity variable, sent to the motors, and of the attractor for this variable. The middle plot shows the respective pan trajectory. In the lower plot, activation of the motor CoS is depicted. Fig. 2 (right) shows the sensorimotor mapping before learning and after several successful saccades. The 4D mapping is shown here as slices along the motor dimensions, arranged in the figure according to the visual dimensions. Before learning, the mapping is initialized as random connections tensor. After each successful saccade, one region in the 4D field, which corresponds to the overlap between activity peaks in the viDNF and miDNF, is updated (one such region is marked with the red circle; note the light-blue dots in the tiles in this region). After only a few successful saccades (nine shown here), a large portion of the 4D space of the mapping is learned (regions marked by the red circle and the red arrows), because of the finite size of activity peaks in the intention DNFs.

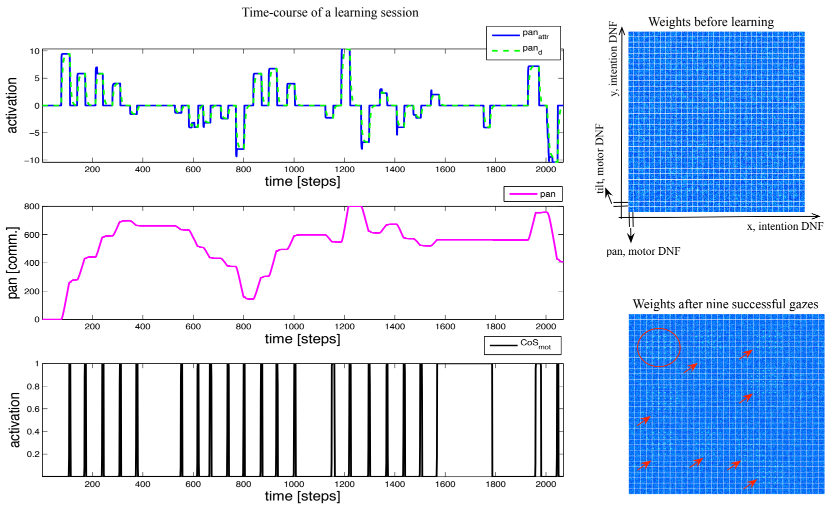


Fig. 2. Left: Exemplary time-course of a learning session. **Right**: The mapping between the visual and the motor intention spaces before learning (top), and after several successful "saccades" (bottom).

4 Discussion

In this paper, we have presented a neural-dynamics architecture that enables autonomous learning of a sensory-motor mapping involved in looking behavior, generated with an eDVS camera mounted on a pan-tilt unit. We have demonstrated how learning accompanies autonomous generation of the looking actions from the low-level sensory input in a closed behavioral loop. We have combined stability of the Dynamic Neural Field representations with elements of the behavioral organization to enable autonomy of the learning process. This includes autonomy of selection of the visual target, initiation of the motor action, termination of the motor action, and decision to trigger the learning dynamics. Such autonomy is critical for implementation of algorithms for adaptation of sensorimotor mappings in real-world robotic scenarios, as well as for understanding autonomy of learning processes in biological cognition.

Acknowledgement. The authors gratefully acknowledge the financial support of DFG SPP *Autonomous Learning*, within Priority Program 1567 and the Telluride Neuromorphic Cognition Engineering Workshop.

References

1. Amari, S.: Dynamics of pattern formation in lateral-inhibition type neural fields. Biological Cybernetics 27, 77–87 (1977)
2. Brooks, R.A.: New approches to robotics. Science 253, 1227–1232 (1991)

3. Conradt, J., Berner, R., Cook, M., Delbruck, T.: An embedded aer dynamic vision sensor for low-latency pole balancing. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 780–785. IEEE (2009)
4. Gaskett, C., Cheng, G.: Online learning of a motor map for humanoid robot reaching (2003)
5. Grossberg, S.: Nonlinear neural networks: Principles, mechanisms, and architectures. Neural Networks 1, 17–61 (1988)
6. Guilherme, G., Araújo, A.F., Ritter, H.J.: Self-organizing feature maps for modeling and control of robotic manipulators. Journal of Intelligent & Robotic Systems 36(4), 407–450 (2003)
7. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics 43(1), 59–69 (1982)
8. Kuperstein, M.: Infant neural controller for adaptive sensory-motor coordination. Neural Networks 4(2), 131–145 (1991)
9. Maes, P., Brooks, R.A.: Learning to coordinate behaviors. In: Proceedings of the Eighth National Conference on Artificial Intelligence, pp. 796–802 (1990)
10. Metta, G., Sandini, G., Konczak, J.: A developmental approach to visually-guided reaching in artificial systems. Neural Networks 12(10), 1413–1427 (1999)
11. Richter, M., Sandamirskaya, Y., Schöner, G.: A robotic architecture for action selection and behavioral organization inspired by human cognition. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS (2012)
12. Ritter, H.J., Martinetz, T.M., Schulten, K.J.: Topology-conserving maps for learning visuo-motor-coordination. Neural Networks 2(3), 159–168 (1989)
13. Saegusa, R., Metta, G., Sandini, G., Sakka, S.: Active motor babbling for sensorimotor learning. In: IEEE International Conference on Robotics and Biomimetics, ROBIO 2008, pp. 794–799. IEEE (2009)
14. Sandamirskaya, Y., Zibner, S., Schneegans, S., Schöner, G.: Using dynamic field theory to extend the embodiment stance toward higher cognition. New Ideas in Psychology. Special Issue "Adaptive Behavior" (in press)
15. Schöner, G.: Dynamical systems approaches to cognition. In: Sun, R. (ed.) Cambridge Handbook of Computational Cognitive Modeling, pp. 101–126. Cambridge University Press, UK (2008)
16. Thelen, E., Smith, L.B.: A Dynamic Systems Approach to the Development of Cognition and Action. The MIT Press, A Bradford Book, Cambridge, Massachusetts (1994)
17. Toussaint, M.: A sensorimotor map: Modulating lateral interactions for anticipation and planning. Neural Comput. 18(5), 1132–1155 (2006), `http://dx.doi.org/10.1162/089976606776240995`
18. Wilson, H., Cowan, J.: A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. Biological Cybernetics 13, 55–80 (1973)
19. Zibner, S.K.U., Faubel, C., Iossifidis, I., Schöner, G.: Dynamic neural fields as building blocks for a cortex-inspired architecture of robotic scene representation. IEEE Transactions on Autonomous Mental Development 3(1), 74–91 (2011)

Learning Temporally Precise Spiking Patterns through Reward Modulated Spike-Timing-Dependent Plasticity

Brian Gardner and André Grüning

Department of Computing, University of Surrey,
Guildford, Surrey, GU2 7XH, United Kingdom
{b.gardner,a.gruning}@surrey.ac.uk

Abstract. Precise neuronal spike timing plays an important role in many aspects of cognitive processing. Here, we explore how a spiking neural network can learn to generate temporally precise spikes in response to a spatio-temporal pattern, through spike-timing-dependent plasticity modulated by a delayed reward signal. An escape noise neuron is implemented as the readout to incorporate the effect of background noise on spike timing. We compare the performance of two different escape rate functions that drive spiking in the readout neuron: the Arrhenius & Current (A&C) and Exponential (EXP) model. Our results show that the network can learn to reproduce target spike patterns containing between 1 and 10 spikes with 10 ms temporal accuracy. We also demonstrate the superior performance of the A&C model over the EXP model for the parameters we consider, especially when reproducing a large number of target spikes.

Keywords: Neuronal Plasticity, Stochastic Neuron, Synapses.

1 Introduction

Generating temporally precise sequences of spikes in response to synaptic input is a fundamental process of neural activity [3]. Learning to generate such responses is considered to take place in the brain through the modification of synaptic strengths between neurons. A variety of synaptic processes are found to drive short to longer term synaptic changes [13]. In particular, Spike-Timing-Dependent Plasticity (STDP) seems to play a key role in learning: a process experimentally observed in hippocampal neurons [2]. A method proposed by Ponulak et al. [15], Remote Supervised learning Method (ReSuMe), demonstrated how such an STDP process can be applied in teaching a network of spiking neurons to map input patterns to arbitrary target spike trains. Furthermore, this method was extended to learning in multilayered networks [10,17]. However, whilst ReSuMe retains a good degree of biological plausibility through the inclusion of STDP, it is uncertain how a feedback signal in the form of a target spike train can realistically be communicated instantaneously during learning.

© Springer-Verlag Berlin Heidelberg 2013

Reward modulated STDP has emerged as a more plausible hypothesis for learning with spiking neurons, where time-dependent correlations in the spiking activity drive synaptic strength modifications, subject to a global reward signal [7,11,12]. For learning a target spike train, this corresponds to strengthening synapses associated with triggering the correct firing times. Previous investigations have demonstrated how networks of spiking neurons learn to reproduce temporally precise sequences of spikes with reward modulation [6,12,8,5]. Furthermore, the importance of background noise has been indicated, where varying neuronal spiking activity is essential for the exploration of reward space towards discovering those spiking patterns that are desired.

In this paper, we implement a stochastic neuron model and investigate its ability to learn a target spike train in response to a spatio-temporal spiking pattern, through a *reward-maximising* STDP rule [8]. For biological plausibility, only delayed reward signals are considered, where feedback on the correctness of a response becomes available upon cessation of the input pattern. We also consider two different forms for the escape rate to drive neuronal spiking and compare their performance over a range of target spike trains.

2 Method

2.1 Single Neuron Model

We consider a two-layer feedforward neural network based on [14,18,9]. A single readout neuron receives input from $1 \leq j \leq M$ presynaptic neurons. The spike train received from the j^{th} neuron is denoted X_j, and the spatio-temporal spiking pattern over all M inputs is $\mathbf{X} = \{X_1, ..., X_M\}$. If the readout neuron generates the output spike train Y in response to $\mathbf{X}$, then its membrane potential at time t is:

$$u(t|\mathbf{X}, Y) := U_{rest} + \sum_{j=1}^{M} w_j \sum_{s \in X_j} \epsilon(t-s) + \sum_{s \in Y} \kappa(t-s) \ , \tag{1}$$

with $U_{rest} = -70$ mV the resting membrane potential and w_j the j^{th} afferent synaptic weight. We approximate the Postsynaptic Potential (PSP) kernel as a double exponential: $\epsilon(s) = \epsilon_0(\mathrm{e}^{-s/\tau_m} - \mathrm{e}^{-s/\tau_s})$, and the reset kernel: $\kappa(s) = \kappa_0 \mathrm{e}^{-s/\tau_m}$; both kernels are set to 0 for $s < 0$. We set $\epsilon_0 = 1.3$ mV, such that a synaptic weight $w_j = 1$ evokes a PSP with an absolute amplitude close to 1 mV, and we set $\kappa_0 = -10$ mV. The membrane time constant is set to $\tau_m = 10$ ms and the synaptic time constant $\tau_s = 0.7$ ms.

To account for background noise, we implement an escape noise model [9]. Spiking events are driven by an escape rate $\rho(u(t))$, that gives the instantaneous firing density for the readout neuron as a function of the time-dependent membrane potential. We set the simulation time step $\delta t = 1$ ms.

In our simulations we consider two different functional forms for the escape rate, the first being the Arrhenius & Current (A&C) [9]:

$$\rho^{\mathrm{A\&C}}(u, \dot{u}) = 2\left(\frac{c_1}{\tau_m} + \frac{c_2}{\sigma}[\dot{u}]_+\right) \frac{\exp\left\{-\frac{[u-\vartheta]^2}{\sigma^2}\right\}}{1 + \mathrm{erf}\left\{-\frac{u-\vartheta}{\sigma}\right\}} \ , \tag{2}$$

with the firing threshold set to $\vartheta = -55$ mV. The parameter σ is the noise amplitude, corresponding to the magnitude in the fluctuations of u due to background stochastic spike arrival. We set $\sigma = 5$ mV, mimicking that measured from *in vivo* experiments [4]. The parameters c_1 and c_2 are set to 0.72 and $\frac{1}{\sqrt{\pi}}$ respectively [9]. The term $[\dot{u}]_+$ indicates that only positive gradients in the membrane potential contribute to the firing density. The error function erf ensures a linear increase in the firing density for $u > \vartheta$.

The second simpler model, referred to as Exponential (EXP), is more commonly used [9]:

$$\rho^{\mathrm{EXP}}(u) = k \exp\{\beta(u - \vartheta)\} \; . \tag{3}$$

We set the stochasticity parameters $k = 0.156$ and $\beta = 0.334$, such that for $u < \vartheta$: $\rho^{\mathrm{EXP}}(u) \approx \rho^{\mathrm{A\&C}}(u, \dot{u} = 0)$, giving comparable levels of noise between the two models.

2.2 Learning Algorithm

A stochastic neuron model allows for the determination of the likelihood for generating the set of output spikes Y in response to $\mathbf{X}$. By the technique of gradient ascent, the eligibility for the j^{th} synapse can be found as [14,7]:

$$e_j(t) = \frac{\rho'(u)}{\rho(u)} \left[\mathcal{Y}(t) - \rho(u)\right] \sum_{s \in X_j} \epsilon(t - s) \; , \tag{4}$$

where $\rho'(u) = \frac{d\rho(u)}{du}$ and $\mathcal{Y}(t) = \sum_{s \in Y} \delta(t - s)$ is the spike train of the readout neuron as a sum of δ functions. Weights are updated as $\dot{w}_j(t) = \eta \mathcal{R} e_j(t)$, with learning rate η and reward signal $\mathcal{R}$. It is unrealistic however, to assume that reward can be delivered instantaneously at every moment in time. Therefore, $e_j(t)$ is low-pass filtered to provide a moving average called the synaptic eligibility trace $E_j(t)$ [7,18], given as:

$$\tau_{\mathcal{R}} \dot{E}_j(t) = e_j(t) - E_j(t) \; , \tag{5}$$

where the time constant $\tau_{\mathcal{R}}$ is matched to the duration of the input pattern $\mathbf{X}$. In our simulations, weights were updated only at the end of each episodic presentation of $\mathbf{X}$ when reward became available, where we set the duration of each episode $T = 500$ ms.

For the A&C model, we determined the eligibility as:

$$e_j^{\mathrm{A\&C}}(t) = \mathcal{A}(u) \left[\mathcal{Y}(t) - \rho^{\mathrm{A\&C}}(u, \dot{u})\right] \sum_{s \in X_j} \epsilon(t - s) \qquad \text{with} \tag{6}$$

$$\mathcal{A}(u) = \frac{2}{\sigma} \left(\frac{1}{\sqrt{\pi}} \frac{\exp\left\{ -\frac{[u - \vartheta]^2}{\sigma^2} \right\}}{1 + \mathrm{erf}\left\{ -\frac{u - \vartheta}{\sigma} \right\}} - \frac{u - \vartheta}{\sigma} \right) \; , \tag{7}$$

where we neglected terms containing higher order time derivatives of $\dot{u}$. Preliminary simulations showed that such contributions were minimal, having little overall impact on learning. For the EXP model, the eligibility is simply given as:

$$e_j^{\mathrm{EXP}}(t) = \beta\, [\mathcal{Y}(t) - \rho^{\mathrm{EXP}}(u)] \sum_{s \in X_j} \epsilon(t - s) \; . \tag{8}$$

2.3 Learning a Target Spike Train

We wish to teach the network to respond to a spatio-temporal spiking pattern $\mathbf{X}$ with an output spike train Y^{out}, matching an arbitrary target spike train Y^{ref}. Similar in approach to [6,5], we use the van Rossum Distance (vRD) [16] to measure the dissimilarity between Y^{out} and Y^{ref}, giving the metric $\mathcal{D}$. We arbitrarily set the coincidence time constant $\tau_c = 15$ ms. To remove the dependence of the vRD on the number of target spikes, $\mathcal{D}$ is normalized by setting $\mathcal{D}_N = \mathcal{D}/\mathcal{D}_0$, where $\mathcal{D}_0$ is the vRD from just Y^{ref}. $\mathcal{D}_N \in [0, \infty)$ is then mapped to a reward value $\mathcal{R} \in (0, 1]$ as $\mathcal{R} = \exp(-\alpha \mathcal{D}_N)$, where we set $\alpha = 4$ such that reward becomes negligible for distances $\mathcal{D}_N > 1$. Maximum reward $\mathcal{R} = 1$ is attained when $\mathcal{D}_N = 0$, corresponding to a perfect match between the spike trains Y^{out} and Y^{ref}. $\mathcal{D}_N$ is determined when the presentation of $\mathbf{X}$ to the network terminates. We additionally set $\mathcal{R} = 0$ when no output spikes are generated, since a lack of firing activity would lead to stagnation in learning.

Rather than directly substitute $\mathcal{R}$ into the weight update rule, we implement an adaptation of the Temporal Difference (TD) error rule, originally defined in classical Reinforcement Learning [1]. Following [6], the TD error on the n^{th} episodic presentation of the input pattern is given as $\delta_{\mathcal{R}}(n) = \mathcal{R}(n) - \langle \mathcal{R} \rangle$, with the moving average of reward updated as $\langle \mathcal{R} \rangle \leftarrow 0.1\mathcal{R}(n) + 0.9 \langle \mathcal{R} \rangle$. The update for the j^{th} synaptic weight after the n^{th} presentation of $\mathbf{X}$ then becomes:

$$\Delta w_j(n) = \eta\, \delta_{\mathcal{R}}(n)\, E_j(T) \; , \tag{9}$$

where $E_j(T)$ is the j^{th} synaptic eligibility trace at the end of the n^{th} episode, with time $t = T$.

2.4 Plasticity Rules

For learning, we implement 'additive' STDP [13], where synaptic weight changes Δw_j are simply clipped if the absolute value $|w_j|$ moves outside of the range $[w_{min}, w_{max}]$. We set $w_{min} = 5 \times 10^{-3}$ and $w_{max} = 5$ as the minimum and maximum attainable absolute synaptic weights respectively. In all cases, plasticity takes place in both excitatory and inhibitory connections, where inhibitory connections have negative values for w_j.

To maintain a homeostatic firing rate and introduce competition between afferent connections, a simplified adaptation of the synaptic scaling rule proposed by [19] is used:

$$\Delta w_j^{scaling} = \gamma\, |w_j|\, \left[N^{ref} - N^{out} \right] \; , \tag{10}$$

where γ is the scaling strength, N^{ref} the number of target spikes and N^{out} the number of spikes generated by the readout neuron over the duration of each learning episode. Weight changes from scaling are implemented 'additively' and take place at the end of each learning episode, where we set the scaling strength $\gamma = 1 \times 10^{-3}$.

2.5 Network Setup and Learning Task

We implemented a two-layer fully-connected feedforward network, consisting of 500 neurons in the first layer and a single readout neuron in the second layer. Either the A&C or EXP model defined the readout neuron. The input pattern $\mathbf{X}$ consisted of an independent Poisson-distributed spike train for every input neuron, each with a mean firing rate of 6 Hz. Synaptic weights between the first layer and readout neuron were initialized by independently selecting each value from a Gaussian distribution, with means 0.32 and 0.26 for A&C and EXP respectively and the standard deviation 1/3 the mean. These values were selected to drive the initial firing rate of the readout neuron to 6 Hz. The ratio of excitatory to inhibitory weights was 4 : 1.

For each learning task the network had to learn to reproduce a target spike train, with spikes selected from a uniform distribution over $[50, T - 50]$. For multiple-spike target trains, target spikes were separated by a minimum of $2\tau_c$ to avoid confliction.

3 Results

We explored the capability of both the A&C and EXP model in learning to reproduce an arbitrary target spike train in response to a fixed input pattern, where the number of target spikes ranged from 1-10. In all simulations we set the learning rate to $\eta = 200$ for both A&C and EXP readout neurons.

To characterise learning, we defined a performance measure p such that desirable responses by the network gave $p = 100\%$ and $p = 0$ otherwise. We considered desirable responses to occur on those episodes where every target spike could be paired to within $\Delta t = 10$ ms of an output spike, given that such values for Δt between output and target spikes had the effect of reducing $\mathcal{D}_N$. We additionally set the constraint that the output spike train must contain the same number of spikes as its target, thereby disallowing spurious spiking. Since there were large fluctuations in the output with each episode, we took the performance as a moving average. The average performance was updated on each episode according to $\tilde{p}(n) = (1 - \lambda)\tilde{p}(n-1) + \lambda p(n)$ with $\lambda = 0.004$. $\tilde{p}(n)$ measured the probability of the network generating a desired response on the n^{th} episode. To measure the convergence in learning, we took a similar approach as Florian [7]: convergence was considered to take place on episode number n_c if $\tilde{p}(n)$ did not become larger than $\tilde{p}(n_c)$ for episode numbers between n_c and $n_c + \nu$. The convergence observation period ν was set between 1000-5000 episodes, scaling with the number of target spikes to be learnt.

Fig. 1 shows the performance and convergence speed of the A&C and EXP escape rates when learning to reproduce 1-10 target spikes. We found that A&C

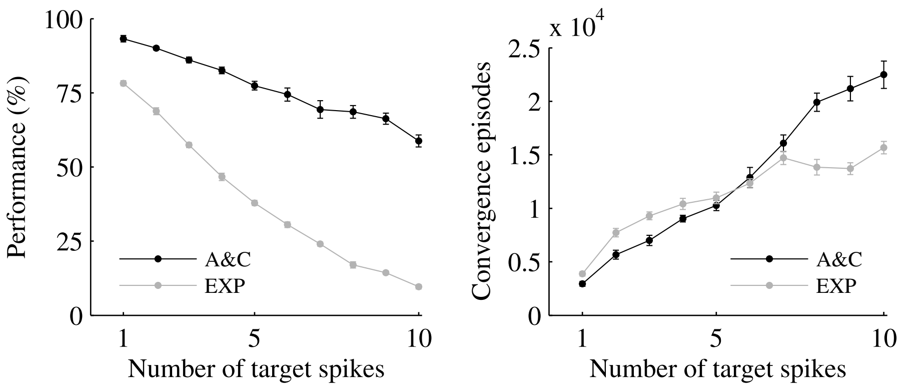


Fig. 1. Each network learning to reproduce an arbitrary N-spike target train in response to a fixed input pattern. (Left) Performance $\tilde{p}$ at convergent episode number n_c for each network. (Right) Number of episodes to convergence n_c in the performance $\tilde{p}$ for each network. Each point is the mean over between 10-20 independent learning tasks, where error bars show the standard error of the mean.

consistently outperformed EXP, where the difference in $\tilde{p}$ between the two models approached 50 percentage points for 10 target spikes. A&C maintained a good level of performance over the entire range of target spike trains we considered, with a minimum of $\tilde{p} = 59{\pm}2\%$ for 10 target spikes. By contrast, the performance of EXP deteriorated rapidly, with $\tilde{p} = 9.6 \pm 0.7\%$ for 10 target spikes. In terms of the convergence speed, A&C converged more rapidly than EXP from between 1 and 5 target spikes, although the reverse was found for target spikes greater than 5, where the convergence time for EXP saturated. For A&C, there was an indication of decreasing convergence rate from 8 target spikes. The decreased number of convergence episodes for EXP reflected the relatively fast attainment of poorer convergent performance.

For illustration, we show a typical spike raster for each escape rate when learning to reproduce an 8-spike target train, shown in Fig. 2. Clearly, A&C outperformed EXP, where we found A&C took just over 7500 episodes to attain a performance of 50%, corresponding to a performance of $\sim$ 10% for EXP. According to Fig. 1, the number of episodes to convergence in $\tilde{p}$ for 8 target spikes were $n_c = (1.99 \pm 0.08) \times 10^4$ and $n_c = (1.38 \pm 0.07) \times 10^4$ for A&C and EXP respectively. In relation to Fig. 2, these values for n_c indicated those episodes beyond which no further gains in performance were possible. Although it might appear that performance converged earlier, especially for A&C, there existed an intermediate period of 'fine-tuning', during which spurious spiking was further reduced to allow for relatively smaller but significant gains in the performance.

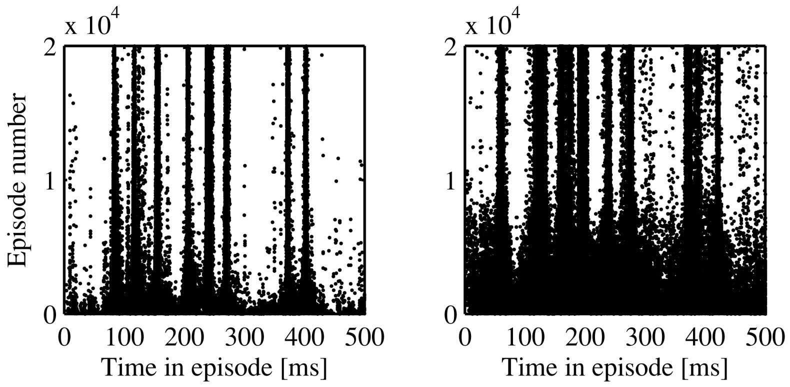


Fig. 2. Learning to reproduce an 8-spike target train in response to a fixed input pattern. Both spike rasters reflect typical network responses. (Left) A&C model. (Right) EXP model. Note the broader spread of the EXP model around the target spike times.

4 Discussion

We have explored the utility of a stochastic neuron model in learning to reproduce temporally precise spiking patterns through reward modulated STDP, a process that might underpin learning in the brain. Furthermore, we have investigated two different escape rate functions to drive neuronal spiking, and compared their performance over a range of target spike trains. We found using an escape noise neuron model to be ideally suited to the task of reproducing target spike trains by reinforcement, given that a degree of background noise was essential in driving explorative spiking during learning. In terms of the escape rate model, A&C performed consistently better than EXP for the set of parameters considered, with the difference in performance being apparent for a larger number of target spikes. We were, however, primarily motivated in applying the A&C rather than just the EXP model given its dependence on the experimentally measurable noise amplitude parameter σ.

In our network we only included one readout neuron for simplicity. More realistically we can expect populations of neurons processing similar input patterns, where the output can be "averaged over" (for example by a leaky integrator) to produce a more deterministic response. Given that our simulations were limited to learning single input-output pattern pairs, future work utilizing such populations would likely facilitate the learning of several such pattern pairs.

Acknowledgement. BG was fully supported by EPSRC grant EP/J500562/1.

References

1. Barto, A., Sutton, R.: Reinforcement learning: An introduction. MIT Press, Cambridge (1998)

2. 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)
3. Bohte, S.: The evidence for neural information processing with precise spike-times: A survey. Natural Computing 3(2), 195–206 (2004)
4. Chance, F., Abbott, L., Reyes, A.: Gain modulation from background synaptic input. Neuron 35(4), 773–782 (2002)
5. El-Laithy, K., Bogdan, M.: A reinforcement learning framework for spiking networks with dynamic synapses. In: Computational Intelligence and Neuroscience 2011, vol. 4 (2011)
6. Farries, M., Fairhall, A.: Reinforcement learning with modulated spike timing dependent synaptic plasticity. Journal of Neurophysiology 98(6), 3648–3665 (2007)
7. Florian, R.: Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. Neural Computation 19(6), 1468–1502 (2007)
8. Frémaux, N., Sprekeler, H., Gerstner, W.: Functional requirements for reward-modulated spike-timing-dependent plasticity. The Journal of Neuroscience 30(40), 13326–13337 (2010)
9. Gerstner, W., Kistler, W.: Spiking neuron models: Single neurons, populations, plasticity. Cambridge University Press, Cambridge (2002)
10. Grüning, A., Sporea, I.: Supervised learning of logical operations in layered spiking neural networks with spike train encoding. Neural Processing Letters 36(2), 117–134 (2012)
11. Izhikevich, E.: Solving the distal reward problem through linkage of stdp and dopamine signaling. Cerebral Cortex 17(10), 2443–2452 (2007)
12. Legenstein, R., Pecevski, D., Maass, W.: A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. PLoS Computational Biology 4(10), e1000180 (2008)
13. Morrison, A., Diesmann, M., Gerstner, W.: Phenomenological models of synaptic plasticity based on spike timing. Biological Cybernetics 98(6), 459–478 (2008)
14. Pfister, J., Toyoizumi, T., Barber, D., Gerstner, W.: Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. Neural Computation 18(6), 1318–1348 (2006)
15. Ponulak, F., Kasinski, A.: Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. Neural Computation 22(2), 467–510 (2010)
16. Rossum, M.: A novel spike distance. Neural Computation 13(4), 751–763 (2001)
17. Sporea, I., Grüning, A.: Supervised learning in multilayer spiking neural networks. Neural Computation 25(2), 473–509 (2013)
18. Urbanczik, R., Senn, W.: Reinforcement learning in populations of spiking neurons. Nature Neuroscience 12(3), 250–252 (2009)
19. Van Rossum, M., Bi, G., Turrigiano, G.: Stable hebbian learning from spike timing-dependent plasticity. The Journal of Neuroscience 20(23), 8812–8821 (2000)

Memory Trace in Spiking Neural Networks

Marta Castellano and Gordon Pipa

Institute of Cognitive Sciences,
University of Osnabrueck, Germany
{mcastellano,gpipa}@uos.de

Abstract. Spiking neural networks have a limited memory capacity, such that a stimulus arriving at time t would vanish over a timescale of 200-300 milliseconds [1]. Therefore, only neural computations that require history dependencies within this short range can be accomplished. In this paper, the limited memory capacity of a spiking neural network is extended by coupling it to an delayed-dynamical system. This presents the possibility of information exchange between spiking neurons and continuous delayed systems.

Keywords: spiking neural networks, memory trace, delayed-dynamical systems, reservoir computing.

1 Introduction

Neurons communicate through action potentials, while the represented cognitive processes operate at slower timescales. The neural system, then, must have some way of storing the short term information required for the cognitive processes.

Classical paradigms for short-term memory state that memory arises through persistent patterns of neural activity, which stabilize through synaptic modification [3]. A relatively new paradigm proposes that spiking neural networks (SNN) encode and store information in their transient dynamics, while computations emerge through the continuous interaction of external stimulus with the internal states of the network [2]. This concept is generalized within the reservoir computing community: any non-linear dynamical system with fading memory, the reservoir, can be used as a computational system that processes stimulus in its transient trajectories [4,5].

If information is retained within neural trajectories, how quickly are those traces forgotten? A spiking neural network, with no slow processes associated, (such as adaptation or synaptic plasticity), has a memory trace of few ms, which is on the same timescale of the intrinsic time constant of single neurons [6]. As a result, computations, that require information to be retained over longer timescales, are not be solvable; for example motor preparation [7,8] or working memory tasks [3].

Several studies have tried to overcome the limited memory trace. Through the addition of feedback connections, [9] brought a generic spiking neural network to develop high-dimensional attractor-like sub-manifolds that solved working

© Springer-Verlag Berlin Heidelberg 2013

memory tasks. Likewise, [10] extended the memory trace by introducing working memory units, neurons connected to the recurrent network by means of plastic synapses that mark the presence of an event as on/off states, bringing the network towards multiple point attractors.

While these two studies propose a model by which the memory trace of the reservoir is extended by the interaction with adjacent neural units, we propose a model by which the memory trace of the reservoir is extended by a non-linear coupling of the network to an external delayed-dynamical system (DDS), which is a general term for a dynamical system that change over time depends on its current and past states.

Our proposal is based on two primary observations. First, delays are ubiquitous in biological systems and including them within mathematical models extend the range of dynamics observed in the system [11]. Second, DDSs can, in a similar fashion, be used as a reservoir to solve computational tasks (first proposed in the PHOCUS European project FP7-ICT-2009-C).

This paper is organized as follows. The method section presents SNN and DDS, with emphasis on the methods of encoding and decoding used to estimate computational performance and the memory trace. In the results section, we first present both SNN and DDS dynamics, together with their computational performance and memory trace. Afterwhich, the results are compared to the case, in which SNN is coupled to DDS, showing that the memory trace of an SNN can be extended by the non-linear coupling with a delayed-dynamical system.

2 Methods

Spiking Neural Network. The spiking neural network (SNN) is modeled with the modeling toolbox CSIM [12]. In short, a set of $N_n = 135$ leaky integrate and fire neurons are placed on a 3x3x15 grid and 20% of them are randomly selected to be inhibitory. The membrane potential V_m^i of a neuron i is described by:

$$\tau_m \frac{dV_m^i}{dt} = -(V_m - E_{rest}) + R_m \cdot (I_{in}^i + I_{syn}^i) \tag{1}$$

with a membrane time constant $\tau_m = 30$ ms, a resting potential $E_{rest} = 0$ and an input resistance $R_m = 1M\Omega$. The spiking threshold is set to 15 mV; the absolute refractory period is 3 ms (excitatory) and 2 ms (inhibitory). The membrane potential is reset to a voltage uniformly drawn from the interval [13.8 mV, 14.5 mV], same values used to initialize V^i each simulation [12]. I_{syn}^i is the sum of recurrent connections currents that arrive at the membrane, while I_{in}^i are the sum of the stimulus currents. Numerical approximation is obtained by the Euler's method with fixed integration step ($\delta t = 0.001$) sec. The neurons are randomly connected with a probability of connection c, which is different among inhibitory (in) and excitatory (ex) neurons: 0.3 ex-ex, 0.2 ex-in, 0.4 in-ex and 0.1 in-in, leading to a total of 2325 synapses.

Delay-Dynamical System. A non-linear system with delayed feedback of the general form $\dot{x}(t) = f(x(t)) + g(x(t-\tau))$, is here named delay-dynamical system (DDS) and implemented by the Mackey-Glass equation [13,14]:

$$\frac{dx(t)}{dt} = \beta \frac{x(t-\tau) + \alpha I_{DDS}(t)}{1 + (x(t-\tau) + \alpha I_{DDS}(t))^n} - x(t) \tag{2}$$

with $\beta = 0.4$ being the coupling factor of the feedback, $n = 1$ the non-linearity exponent, $\alpha = 0.05$ the history dependence, $\tau = 80$ the delay time and $I_{DDS}(t)$ receives the external stimuli. Numerical approximation of the delayed differential equation is obtained by the Heun's method with fixed integration step ($h = 0.001$).

Stimulus to the Reservoir. The reservoir receives $R = 2$ dimensional time-varying stimulus r_1 and r_2, an non-homogeneous Poisson process with uniformly distributed rates $\lambda = [10, 20, 40]$ Hz, so that at each point in time t the Poisson process is drawn from $r_i(t) \approx \lambda(t) \cdot \delta t$ with $\lambda(t)$ being uniformly distributed from the set $[10, 20, 40]$, see Figure 2 A).

Encoding the Stimulus. The DDS encodes stimulus as follows: the delay term τ is divided into M virtual nodes, i.e., a set of M points equidistant distributed over the delay τ, see Figure 1. The virtual nodes are those time points that encode the stimulus $r_i(t)$. The stimulus $r_i(t)$ is previously preprocessed, referred as the masking procedure. Masking can be seen as a multiplication of the stimulus $r_i(t)$ with the masking function $m(t)$, so that the stimulus that each of the virtual nodes receive $I_{DDS}(t)$, is defined as $I_{DDS}(t) = \sum_{i=1}^{R} r_i(t) \cdot m(t)$, where $m(t)$ is a binary random vector so that $m(t) \in [-1, 1]^M$. Masking the stimulus has three goals: to multiplex the stimulus over the delay line, to ensure that every virtual node of the delayed line receives a linear combination of different dimensions of the stimulus R and ensures that the delayed dynamical system is constantly in a transient regime. Furthermore, the encodign of the stimulus on the delay line is modulated by the parameter k, here called encoding scheme, which reflects the number of time steps t of the stimulus that are going to be encoded within a delay line. In short, a delay line encodes $1/k$ time steps of the stimulus and a time step t is projected onto $N_{in} = 200$ virtual nodes. In this way, the total number of virtual nodes within a delay line scales by the parameter k so that $M = \frac{1}{k} \cdot N_{in}$.

The SNN encodes external stimulus by means of spike trains. The stimulus $r_1(t)$ and $r_2(t)$ are time-varying firing rates from which Poisson spike trains are drawn, and each of them is mapped to an independent subset of 8 neurons $j \in N_n$. The spike probability on the interval δt is given by $p(spike = 1(t - \delta t, t + \delta t)) = r_i(t)\delta t$. The spike trains are converted into currents by the convolution of the spikes with an exponential decay term, so that the current I_{in} resembles an EPSP. Specifically $I_{in} = W * e^{-1/\tau_s}$, where $\tau_s = 4$ is the decay time of the EPSP and $W = 0.15$ scales the EPSP amplitude.

Decoding the Stimulus. The reservoir activation $z(t)$, which can be either the SNN or the delayed-dynamical system, for N nodes and total simulation time Q is denoted $\mathbf{A}$, and the expected output signal of the reservoir (the target signal to be approximated) as $y(t)$ for $t \in (t_0, ..., t_Q)$. The aim of the linear regression (i.e. maximum likelihood with normal distributed residuals) is to find the set of weights w that fulfill $y = wA$, obtained by applying the pseudo-inverse, so that $w = y(\mathbf{A})^{-1}$. The target signal is defined in this paper as the sum of the two-dimensional stimulus, so that $y(t) = r_1(t) + r_2(t)$. Learning the weights w is denoted as the training phase. Next, in the testing phase, the weights w are kept fixed and an output $u(t)$ is obtained from the network activity,so that $u(t) = wA$. Finally, the accuracy of the linear regression is evaluated on the testing set as described in the computational performance section. Intuitively, the weights w of the readout can be trained in a task specific way, so that for every task, there is a linear combination of nodes in the reservoir that can be used to approximate the target signal $y(t)$.

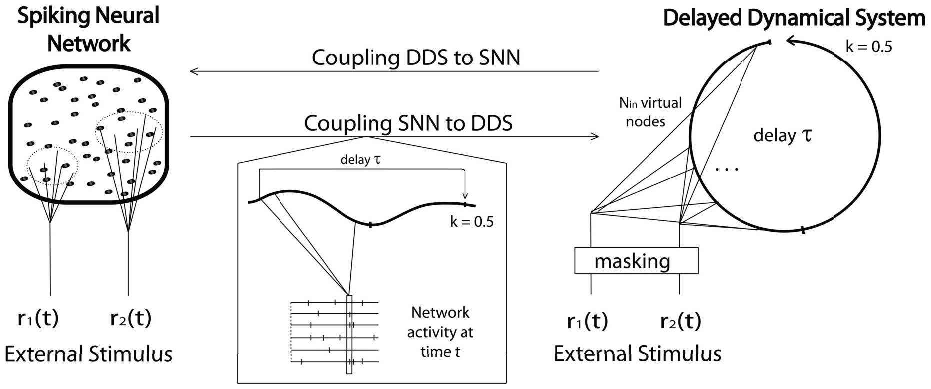


Fig. 1. Visualization of the non-linear coupling between DDS and SNN

Computational Task. The task of the reservoir consists in reconstructing a time-dependent stimulus $r_i(t)$ by reading out the activity of the reservoir at later times t_{lag}.

Computational Performance and Memory Trace. Computational performance $CP(t)$ is defined as the correlation coefficient between target $y(t)$ and estimated output $u(t_{lag})$ at time $t = t_{lag}$, so that $CP(t_{lag}) = corr(y(t), u(t_{lag}))$. Memory trace MT is defined as the maximum time at which the input can be decoded from the network with a correlation coefficient higher than 0.5, so that $MT = max\ \rho_i$, where ρ_i is the time lag t_{lag} at which $CP(t_{lag})$ becomes lower than 0.5.

Coupling between Delayed-Dynamical System and Spiking Neural Network. The non-linear coupling can be visualized in Figure 1. The SNN encodes the signal of the DDS, $x(t)$, by means of 16 analog synapses, represented in the I&F neurons as a current in the term $I_{in} = W_{DDS} \cdot x(t)$, where $W_{DDS} = 0.01$.

3 Results

This section is divided in two different parts. First, we characterize both SNN and DDS dynamics and present its computational performance and memory trace, providing a qualitative and quantitative description of the two models. Second, we compare the results to the case in which SNN is coupled to DDS, showing that the memory trace of an SNN can be extended by the non-linear coupling with a delayed-dynamical system.

3.1 Memory Trace of SNN and DDS

The dynamic responses of the two systems when processing the stimulus are presented in Figure 2 B) and Figure 2 C). The system operates in a fixed point regime (i.e. single fixed point, Figure 2 D, left), perturbed by external stimulus (see Figure 2 D, right).

The computational performance at different time lags and the memory trace of the SNN and DDS are quantified in Figure 2 E). The simulated SNN (parameter specification in methods section) has a maximum computational performance of 0.9 at time lag zero and a memory trace of 0.11 sec, consistent with results presented in [1].

The memory trace and computational performance of the DDS varies together with the variation of the parameter k, which controls the projection of the stimulus to the DDS (encoding scheme). At $k = 1$, $\delta t = 1$ ms of stimulus $r_i(t)$ is mapped to the $M = 200$ virtual nodes contained in a delayed loop $\tau = 80$. In this encoding scheme, the DDS has a maximum computational performance of 0.87 at time lag zero and a memory trace of 0.067 sec. Increase of k leads to the increase of the time steps δt of the stimulus that are mapped within the delay line. The higher the k, the longer the memory trace of the DDS. Consider for example the case where $k = 0.04$, where a single delay loop encodes 25 δt of the stimulus in a total of 5000 virtual nodes (note that the number of virtual nodes increases as $1/k \cdot N_{in}$). This encoding scheme has a maximum computational performance of 0.93 at time lag zero and a memory trace of 0.27 sec.

The increase in memory trace observed by the increase of k cannot be explained by the increase on the number of virtual nodes: a DDS with an encoding parameter $k = 1$ and $M = 5000$ virtual nodes has a maximum computational performance of 0.9 at time lag zero and a memory trace of 0.07 sec. The results obtained in this section are used as a baseline to compare the computational performance and memory trace of the following simulations.

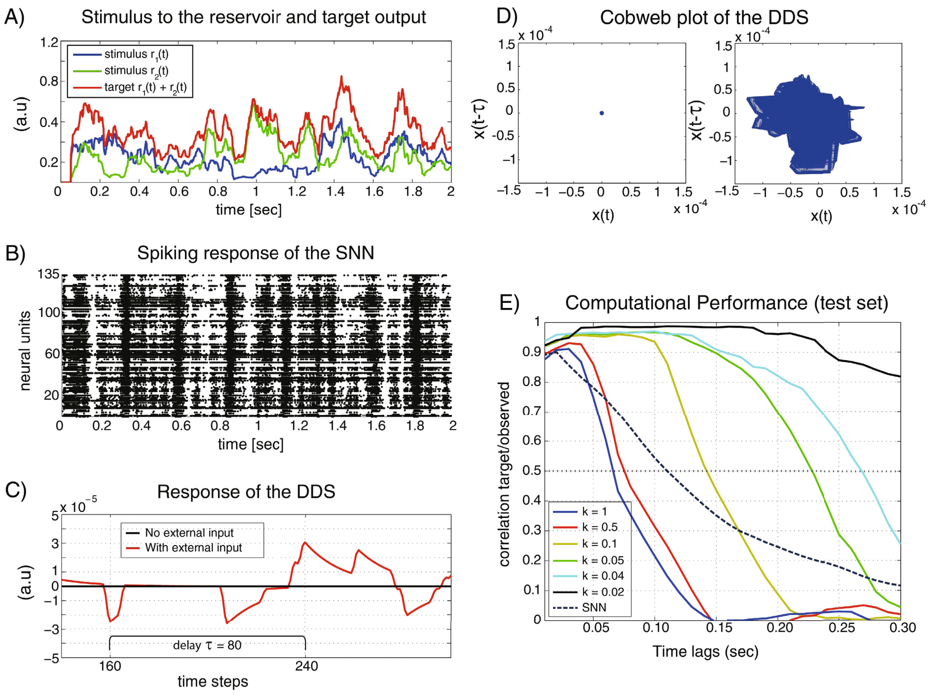


Fig. 2. A) Stimulus and target output for the reservoir. B) Spiking response of the SNN. C) Response of the DDS within a delay line (see equation 2). D) Cobweb plot of the DDS (left) and the DDS receiving external stimulus $r_1(t)$ and $r_1(t)$ (right). E) Estimation of the computational performance at different time lags for the SNN *(dotted blue line)* and the DDS, where k changes the encoding to the DDS.

3.2 Memory Trace of SNN Coupled to DDS

This section aims to estimate whether the SNN shows an increased memory trace when coupled to the DDS. Accordingly, we performed two different simulations: on the one hand, the SNN receives input from the DDS, and on the other hand, the SNN receives input from both DDS and external stimulus.

In the first case, the DDS encodes stimulus r_1 and r_2 and the signal from the DDS is sent to the SNN. With this, we test whether the SNN extracts information from the stimulus that is being processed by the DDS. In this case, the computational performance of the SNN at zero time lag is lower compared to the baseline, while the memory trace of the SNN is longer than baseline as long as the DDS encodes the input at $k > 0.5$ (for instance, at $k = 0.04$ $CP(t_{lag=0}) = 0.62$ and $MT = 0.16$ sec).

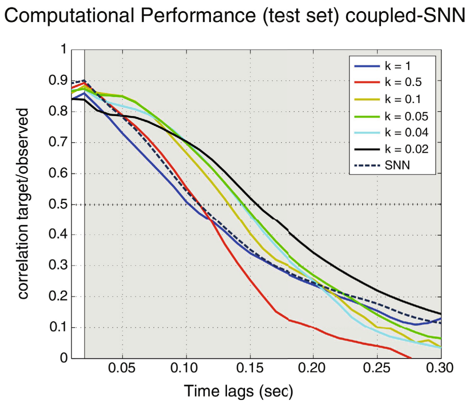


Fig. 3. Computational performance and memory trace of the coupled-SNN for different k, compared to the non-coupled SNN *(dotted line)*

In the second case, the SNN receives stimulus from both DDS and external stimulus (r_1 and r_2), referenced as coupled-SNN. With this simulation we aim to estimate whether the SNN can integrate DDS signals to the ongoing stimulus processing. Results presented in Figure 3 show that the coupling to the DDS does not change the computational performance of the SNN at time lag zero. Nevertheless, as long as the DDS encodes stimulus at $k > 0.5$, the coupled-SNN has longer memory trace than the baseline SNN (0.11 sec baseline memory trace versus 0.14 sec at $k = 0.04$). The gray shadow in Figure 3 indicates that the difference to the non-coupled SNN is statistical significant ($p = 1.7 \cdot 10^{-16}$, one way ANOVA for 20 observations).

4 Discussion

Previous models for short-term memory in spiking neural networks (SNN) proposed that low-dimensional attractors in the circuit dynamics store stimulus information [3]. Here we explored a new paradigm, whereby short-term memory is implemented in the transient dynamics. Within this framework, the memory trace is limited by the length of the neural trajectory, modulated by features such as intrinsic neuron time constants [6] or network topology [15]. We propose a modification of the framework by which a SNN extends its memory trace by a non-linear coupling with a delayed-dynamical system (DDS).

As a proof of principle, we defined a generic DDS and proposed a non-linear coupling with the SNN, which lead to the increase of the memory trace of the spiking neural network. This highlights an essential feature: including delayed coupling within a spiking neural network extended the memory trace, which could not be maintained by the spiking network alone.

The relevance of this finding relies upon neural systems being delayed dynamical systems. Often spiking neuron models are simplified up the point where delays play no role. The addition of delays, not only increase the dynamic range of mathematical models [11], but also increases the range of timescales at which the system processes and retains stimulus.

Acknowledgments. This research was partially supported by the EU grant PHOCUS European project FP7-ICT-2009-C. The authors declare that there is no conflict of interests.

References

1. Maass, W., Natschläger, T., Markram, H.: Fading memory and kernel properties of generic cortical microcircuit models. Journal of Physiology 98, 315–330 (2004)
2. Buonomano, D.V., Maass, W.: State-dependent computations: spatiotemporal processing in cortical networks. Nature Reviews Neuroscience 10, 113–125 (2009)
3. Durstewitz, D., Seamans, J.K., Sejnowski, T.J.: Neurocomputational models of working memory. Nature Neuroscience 3(suppl.), 1184–1191 (2000)
4. 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)
5. Jäger, H.: The echo state approach to analysing and training recurrent neural networks. GMD Report 147 (2001)
6. Mayor, J., Gerstner, W.: Signal buffering in random networks of spiking neurons: Microscopic versus macroscopic phenomena. Physical Review E 72, 15 (2005)
7. Körding, K.P., Wolpert, D.M.: Bayesian integration in sensorimotor learning. Nature 427, 244–247 (2004)
8. Churchland, M.M., et al.: Neural population dynamics during reaching. Nature 487, 51–56 (2012)
9. Maass, W., Joshi, P., Sontag, E.D.: Computational aspects of feedback in neural circuits. PLoS Comput. Biol. 3, e165 (2007)
10. Pascanu, R., Jäger, H.: A Neurodynamical Model for Working Memory. Neural Networks 1, 123 (2010)
11. Forde, J.E.: Delay Differential Equation Models in Mathematical Biology. PhD Thesis
12. Natschläger, T., Markram, H., Maass, W.: Computer Models and Analysis Tools for Neural Microcircuits. Neuro- Science Databases. A Practical Guide, 121–136 (2003)
13. Mackey, M.C., Glass, L.: Oscillation and Chaos in Phisiological Control Systems. Science (1977)
14. Appeltant, L., et al.: Information processing using a single dynamical node as complex system. Nature Communications 2, 468 (2011)
15. Ganguli, S., Huh, D., Sompolinsky, H.: Memory traces in dynamical systems. PNAS 105, 18970–18975 (2008)

Attention-Gated Reinforcement Learning in Neural Networks—A Unified View

Tobias Brosch, Friedhelm Schwenker, and Heiko Neumann

Institute of Neural Information Processing, University of Ulm,
89069 Ulm, Germany
{tobias.brosch,friedhelm.schwenker,heiko.neumann}@uni-ulm.de

Abstract. Learning in the brain is associated with changes of connection strengths between neurons. Here, we consider neural networks with output units for each possible action. Training is performed by giving rewards for correct actions. A major problem in effective learning is to assign credit to units playing a decisive role in the stimulus-response mapping. Previous work suggested an attentional feedback signal in combination with a global reinforcement signal to determine plasticity at units in earlier processing levels. However, it could not learn from delayed rewards (e.g., a robot could escape from fire but not walk through it to rescue a person). Based on the AGREL framework, we developed a new attention-gated learning scheme that makes use of delayed rewards. Finally, we show a close relation to standard error backpropagation.

Keywords: Reinforcement Learning, Backpropagation, AGREL, Delayed Rewards, Attention, Hebbian Learning.

1 Introduction

When investigating the plasticity of connection weights in a neural network in tasks involving a reward signal, reinforcement learning was shown to be an effective tool. Depending on the underlying methodology its biological plausiblity varies [16,14]. The proposed mechanisms range from detailed biological models resembling functionalities of the basal ganglia [14,7,1,5] over models using Hebbian plasticity in neural networks [9,10,11] to machine learning models using error backpropagation (BP) in neural networks [15,13,2,3]. While the latter are very effective, they lack biological plausibility. Here, we focus on the model of [10] utilizing a biologically plausible global learning signal with Hebbian plasticity of connection weights that is gated by an attentional feedback signal to increase plasticity of task relevant-neurons. This attention-gated reinforcement learning (AGREL) [10] is, however, limited to tasks with immediate reward delivery. We propose a novel learning scheme extending AGREL to also incorporate delayed rewards (e.g., a robot walking through fire to rescue a person). Similarly to [11], it also extends AGREL by learning an action-value function in the output layer. We demonstrate that the new model also improves learning in the reduced case of immediate rewards as studied in [10], and show a close relation to BP for learning the connection weights in the employed network.

© Springer-Verlag Berlin Heidelberg 2013

2 Task and Network Design

We employ a three-layer neural network shown in Fig. 1 to simulate the selection of one of C mutually exclusive actions depending on the presented state s_t (c.f. [10]). The state is represented in the input layer with activities $\{X_i\}_{i=1\dots N}$. Connections weighted by v_{ij} propagate activity to a hidden layer with activities $\{Y_j\}_{j=1\dots M}$. Post-synaptic activity depends on a nonlinear transfer function g (we employ a logistic function like in [10]). Weights w_{ij} propagate activity from

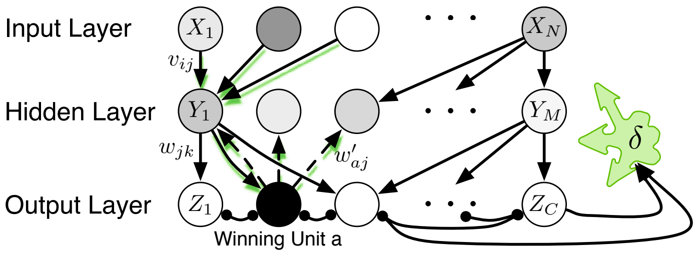


Fig. 1. Three-layer neural network. An input pattern represented by N activities X_i is propagated via weights v_{ij} to M hidden layer neurons of activities Y_j (solid arrows). Weights w_{jk} propagate activity to the output layer with activities Z_k in which neurons engage in a competition (circular arrow heads). Activity of the winning unit a is propagated back to the hidden-layer by feedback weights w'_{aj} (dashed arrows). Learning is initiated by a global learning signal δ.

the hidden to the output layer. An essential part of AGREL is that neurons in the output layer engage in a competition such that only one of C output neurons gets activity $Z_a = 1$, where the probability depends on the presynaptic activity:

$$Y_j = g(Y_j^{pre}) \,, \quad \text{with} \quad Y_j^{pre} = \sum_{i=0}^{N} v_{ij} X_i \,, \quad g(x) = \frac{1}{1+\exp(-x)} \,, \tag{1}$$

$$\mathbb{P}(Z_k = 1) = \frac{\exp(Z_k^{pre}/\tau)}{\sum_{k'=1}^{C} \exp(Z_{k'}^{pre}/\tau)} \,, \quad \text{with} \quad Z_k^{pre} = \sum_{j=0}^{M} w_{jk} Y_j \,. \tag{2}$$

Activity of the winning neuron can be propagated backwards by feedback weights w'_{aj} to gate synaptic plasticity of task relevant neurons (attention gating) in earlier layers. In order to incorporate delayed rewards (in contrast to [10]), we utilize the presynaptic activity Z_k^{pre} of the neural network output layer as function approximator for an action-value function $Q_\pi(s,a) = Z_a^{pre}(s)$. This maps a state s and action a to the expected discounted return

$$\mathbb{E}_\pi\{R_t | s_t = s, a_t = a\} \,, \quad R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,, \tag{3}$$

where π is the current policy of the agent (with $\pi(s,a)$ denoting the probability of selecting action a in state s), r_t is the reward the agent receives at time t and and $0 \leq \gamma \leq 1$ is a discount factor (for γ close to 1, the agent becomes more farsighted). The goal is to find an optimal policy π^* that optimizes the expected discounted return. The corresponding optimal action-value function is denoted $Q^*(s,a)$. To deal with delayed rewards, we will employ a Sarsa-style learning signal from temporal difference (TD) methods for control problems. It encodes the difference between the improved discounted return expectation (based on the experienced reward) and the currently expected discounted return (see e.g. [13] for more details). Because of the combination of a Sarsa-style learning signal and AGREL, we will name our novel extended learning scheme SAGREL.

2.1 Learning in SAGREL

Neural plasticity in SAGREL—as well as in AGREL—depends on two signals that jointly determine plasticity: First, the global reinforcement error signal δ and second, an attentional feedback signal originating from the winning neuron in the output layer and delivering a feedback signal along the connection of the reverse hierarchy network to guide attentional resources. To incorporate learning of delayed rewards, we adapt the definition of the global error signal δ. The learning signal δ in AGREL is only defined for immediate rewards as (c.f. [10])

$$\delta_{AGREL} = \begin{cases} 1 - \mathbb{P}(Z_a = 1)\,, & r = 1\,, \quad \text{in successful trials,} \\ -1\,, & r = 0\,, \quad \text{otherwise.} \end{cases} \tag{4}$$

In SAGREL we adopt a TD Sarsa-style learning signal (similarly to [11]) that directly depends on the presynaptic activity of the output layer $Z_a^{pre}(s) = Q(s,a)$

$$\delta_{SAGREL} = \left[r_{t+1} + \gamma Z_{a_{t+1}}^{pre}(s_{t+1}) \right] - Z_{a_t}^{pre}(s_t)\,, \tag{5}$$

where $r_{t+1} \in \mathbb{R}$ is the reward the agent receives after state s_t taking action a_t, experiencing state s_{t+1} and choosing action a_{t+1} to perform in the next step (see e.g. [13] for more details on Sarsa). Learning of delayed rewards is achieved by incorporating the minuend $r_{t+1} + \gamma Z_{a_{t+1}}^{pre}(s_{t+1})$ which provides a more accurate measure of the expected discounted return (because it incorporates the experienced reward r_{t+1}) than the current estimation $Z_{a_t}^{pre}(s_t)$. We adopt the feedback signal and the neural plasticity mechanism of AGREL as described in [10] which we briefly summarize in the following.

Weights are changed according to a variant of an Hebbian rule, where changes depend upon the product of pre- and postsynaptic activity [8,6]. Weight changes between the hidden and output layer are described by

$$\Delta w_{jk} = \beta \cdot f(\delta) \cdot Y_j \cdot Z_k\,, \tag{6}$$

where β is the learning rate and f is an expansive function increasing the effective learning rate for unexpected rewards. In AGREL f is defined as $f(\delta) = \delta/(1-\delta)$

for $\delta \geq 0$ and $f(\delta) = -1$ otherwise. In SAGREL f needs to be adjusted to the range of the rewards of a given task but can also be generically set to $f(\delta) = \delta$, the identity function. Note that $Z_a = 1$ and $Z_k = 0$ for $k \neq a$, such that only connections w_{ja} to the winning neuron are adjusted. Weight changes between the input and the hidden layer are also governed by Hebbian plasticity but additionally gated by the factor fb_{Y_j} representing feedback of the winning unit:

$$\Delta v_{ij} = \beta \cdot f(\delta) \cdot fb_{Y_j} \cdot X_i \cdot Y_j \,, \qquad fb_{Y_j} = (1 - Y_j) \cdot \sum_{k=1}^{C} Z_k \cdot w'_{kj} \,, \quad (7)$$

where the factor $(1 - Y_j)$ reduces the effect of feedback on the plasticity of the connection weights of highly active units. After the competition $Z_a = 1$ and $Z_k = 0$ for $k \neq a$ and the equation reduces to

$$\Delta v_{ij} = \beta \cdot X_i \cdot Y_j \cdot w'_{aj} \cdot f(\delta) \cdot (1 - Y_j) \,. \quad (8)$$

Cortical anatomy and neurophysiology suggests that feedforward and feedback connections are reciprocal [4,12]. Thus, we set $w_{jk} = w'_{kj}$ (see [10], their sect. 5.4 for a study about deviations from exact reciprocity).

2.2 Comparison between AGREL and SAGREL

In the reduced case of single step tasks and immediate reward delivery (as studied in [10]) δ_{SAGREL} reduces to $r - Z_a^{pre}$ because the expected reward for the next state $Z_{a_{t+1}}^{pre} = 0$ (epsiode ends after the first state). In rewarded trials $(r = 1)$ this reduces to $\delta_{SAGREL} = 1 - Z_a^{pre}$ which is qualitatively the same as $\delta_{AGREL} = 1 - \mathbb{P}(Z_a = 1)$ because $\mathbb{P}(Z_a = 1)$ is a monotonically increasing function of Z_a^{pre}. In unrewarded trials $(r = 0)$, $\delta_{AGREL} = -1$ causes weights to potentially diverge to $-\infty$ because exploration keeps the agent encountering each state in an infinite amount of trials. In SAGREL $\delta_{SAGREL} = -Z_a^{pre}$ which for $Z_a^{pre} \geq 0$ is qualitatively identical to AGREL but prevents weights from approaching $-\infty$ because for $Z_a^{pre} < 0$, δ_{SAGREL} becomes positive. To summarize, the novel SAGREL can deal with delayed rewards tasks while maintaining the computational complexity and allowing an unlimited range of rewards, i.e. $r \in \mathbb{R}$.

2.3 Comparison to Standard Backpropagation in Function Approximation Using Layered Neural Networks

We will now compare the attention-gated reinforcement mechanisms (see previous sections) against standard BP. We utilize the same network as sketched in Fig. 1 but in standard BP, the competition is not modelled in the final layer but carried out separately. Similarly to SAGREL, we interpret the presynaptic activity Z_k^{pre} as function approximation of an action-value function $Q(s, a) = Z_a^{pre}$. For now we assume knowledge of the optimal action-value function Q^*. A commonly used L^2 error function is (c.f. [13])

$$E = \frac{1}{2}\delta^2 = \frac{1}{2}\left(Q^*(s_t, a_t) - Q^{\pi_t}(s_t, a_t)\right)^2 = \frac{1}{2}\left(Q^*(s_t, a_t) - Z_{a_t}^{pre}(s_t)\right)^2 \,. \quad (9)$$

The derivative with respect to a parameter vector $\boldsymbol{\theta}$ (here $\boldsymbol{\theta} = (\boldsymbol{v}, \boldsymbol{w})$ the concatenation of the weights v_{ij} and w_{jk}) is then given by

$$\nabla_{\boldsymbol{\theta}} E = \left(Q^*(s_t, a_t) - Z^{pre}_{a_t}(s_t)\right) \nabla_{\boldsymbol{\theta}} Z^{pre}_{a_t}(s_t) = \delta \cdot \nabla_{\boldsymbol{\theta}} Z^{pre}_{a_t}(s_t) \,. \tag{10}$$

Approximating Q^* by the one-step Sarsa-style return, $r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) = r_{t+1} + \gamma Z^{pre}_{a_{t+1}}(s_{t+1})$, results in the following gradient descent update in the three-layer network in Fig. 1:

$$\Delta w_{jk} = -\beta \cdot \frac{\partial E}{w_{jk}} = \begin{cases} \beta \cdot \delta \cdot Y_j(s)\,, & k = a \\ 0\,, & k \neq a \end{cases} \tag{11}$$

$$\Delta v_{ij} = -\beta \cdot \frac{\partial E}{v_{ij}} = \beta \cdot \delta \cdot w_{ja} \cdot g'\left(Y^{pre}_j(s)\right) \cdot X_i(s)\,, \tag{12}$$

where $\delta = r_{t+1} + \gamma Z^{pre}_{a_{t+1}}(s_{t+1}) - Z^{pre}_{a_t}(s_t)$ and g' denotes the derivative of the transfer function g. We can now compare the weight updates of SAGREL and BP. For weight updates between hidden and output layer, Δw_{jk}, we compare eqns. (11) and (6) which are identical if and only if $f(\delta) = \delta$, the identity function. For weight updates for connections between input and hidden layer, Δv_{ij}, we compare eqns. (12) and (8). When g is the logistic function, eqn. (1), $w_{ja} = w'_{aj}$ and $f(\delta) = \delta$, the identity function, we note that both equations are identical. This is a remarkable result with respect to two properties. *First,* the result implies that BP results in biologically plausible Hebbian plasticity in a paradigm where the teacher signal is replaced by a reward deployment mechanism. *Second,* attention-gated reinforcement learning employs the same efficient updates as the mathematically motivated gradient descent BP. This also explains the empirical observations of [10] that the average weight changes of AGREL are identical to those of BP for instantaneous learning situations. Additionally, this result also extends to [11] in the special case, when the tags (which mark previously active and task relevant neurons) immediately decay (i.e. their $\lambda\gamma = 1$).

3 Experimental Comparison of Modells

First, we compare AGREL with SAGREL in two different experiments presented in [10]. Second, we demonstrate that SAGREL is capable to learn a delayed rewards task. Similarly to [10], we equipped the network with additional degrees of freedom by adding bias neurons with activities $X_0 = Y_0 = 1$, initialized connection weights with values drawn uniformly from the interval $[-0.25, 0.25]$ and used the same performance criterion. The criterion for a successful learning episode was that the probability of correct classification was at least 75% for each of the input patterns.

3.1 Immediate Reward Tasks

Two tasks were used to compare the performance levels to the results in [10].

Exclusive-Or (XOR) Task. The exclusive-or task presents binary input patterns to two input units, namely 01, 10, 11 and 00. The agent must activate one of two output units with the first one being active for $01, 10$ and the second one being active for $11, 00$. The number of hidden neurons was 2 or 3.

Counting Task. In this task, the network must determine how many of N input neurons are switched on. Consequently, the number of output units is $N+1$ (also the number of hidden units). In one iteration of a learning trial all of the $N+1$ different stimulus classes are presented. The specific units that are switched on in each presentation are determined randomly (identical as in [10]).

3.2 Comparison of AGREL and SAGREL

Table 1 displays the minimal median number of iterations (one iteration equals the presentation of all input patterns) required to reach the criterion when optimizing with respect to learning rate β (data for AGREL from [10]) for both models. To compare our method to the results presented in [10], τ was set to one. The data show that SAGREL indeed outperforms AGREL in both tasks (though we used the simplest choice for the function f, the identity function). Based on the investigation outlined in sect. 2.2, this is expected since SAGREL utilizes a more distinct error signal. This demonstrates that our extended learning scheme not only incorporates learning of delayed rewards but also improves the performance in the reduced case studied in AGREL.

Table 1. Comparison of AGREL and SAGREL in the XOR and Counting task (smaller number of iterations is better). Values for AGREL are from [10]. Presented is the median number of iterations (from 500 separate trials) required to reach the 75% performance criterion optimized with respect to β. In SAGREL we determined the minimal number of iterations by performing a grid-search for values of β in $[0.3, 1]$ in steps of 0.025. The data show that the new learning scheme (SAGREL) constantly outperforms the original implementation (AGREL).

Task	AGREL [10]		SAGREL (new)	
	Iterations	β	Iterations	β
XOR (2 hidden units)	535	0.35	422	0.475
XOR (3 hidden units)	474	0.45	297	0.6
Counting 2 inputs	157	0.4	30	0.875
Counting 3 inputs	494	0.25	148	0.325
Counting 4 inputs	1316	0.1	485	0.2

3.3 Delayed Rewards Task

To confirm that SAGREL can indeed learn tasks incorporating delayed rewards, we tested it in a simple task. In this task, N input and output units are used. It always starts with the first unit being active. In a correct trial, the agent selects the same output neuron a as the active input neuron and experiences the next state in which neuron $a+1$ is active (only one input neuron is active at a time). The agent always receives a negative reward, $r = -0.1$, except for the last state, when neuron $a = N$ is active and the agent selects neuron a.

In this case the agent is rewarded with $r = 1$. Whenever a wrong action is selected, the task is reset to the beginning. An episode ends, when the agent gains a positive reward or exceeds a maximum of 30 trials in trying so. Learning was performed for $\tau = 2$ that was multiplied with $\tau_D \leq 1$ after each episode to make the policy a bit more greedy. The employed performance criterion was that the greedy policy with respect to the current action-value function represented by the network (i.e. $\tau = 0$) gained the maximum reward for 50 iterations in a row. To identify the parameters that gain a minimal number of iterations to reach the performance criterion (we measure the median number of iterations in 500 learning task repetitions), we performed a grid search for the parameters $\beta \in [0.6, 1]$ in steps of 0.025 and $\tau_D \in [0.92, 1]$ in steps of 0.005. For $N = 3$, the SAGREL algorithm needed a minimum median number of 190 iterations for $\beta = 0.7$ and $\tau_D = 0.94$ (note that AGREL is not able to solve this task because it cannot deal with delayed rewards). In comparison, a Q-table Sarsa implementation needed only 55 iterations for $\beta = 0.625$ and $\tau_D = 0.93$. This disadvantage can be accounted to the fact that the task is much easier for a Q-table implementation, because there are only N different input states (in contrast to an infinite number of possible input states in SAGREL).

To summarize, SAGREL can indeed learn tasks of delayed rewards and, as such, extends the functionality of the AGREL framework.

4 Discussion

We presented a novel reinforcement learning algorithm for a three-layer neural network to choose one of a finite number of actions based on an infinite number of possible input states. The novel model we propose is based upon the work by Roelfsema and colleagues in [10] and extends it to deal with the important group of delayed rewards tasks. Similarly to [11], it learns Q-values in the output layer. Unlike [11], this paper focuses on the Hebbian nature of the learning dynamics and the connection to standard backpropagation (BP) in reinforcement learning. To do so, we did not explicitly model the transient neuron type used in [11] (which does not change the underlying theoretic comparison to BP but makes it harder to identify). As shown in [10], the model can easily be extended to actor-critic architectures and more than three layers. Our theoretical results lead to the prediction, that the novel learning scheme (and thus also [11]) is superior to the previous framework presented in [10] which we confirmed in two different experiments. Finally, we demonstrated that attention-gated reinforcement learning is closely related and in some cases formally identical to standard error backpropagation in the same neural network architecture. This is a remarkable result since this proves that the biologically plausible mechanisms employed here result in the same effective neural plasticity rule that aims at optimizing an error energy function as in the mathematically motivated BP learning scheme. Additionally, this connection proves that whenever the network (seen as a function) can describe the mapping of states to action values, SAGREL will succeed in finding at least a local solution to the error minimization problem given small enough learning rates.

Acknowledgments. We thank P.R. Roelfsema for his helpful comments. TB is supported by the Graduate School MAEIC at Ulm University and FS and HN are supported in part by the SFB/TRR 62 funded by the DFG. HN was further supported by a grant from the German BMBF, project 01GW0763.

References

1. Brown, J., Bullock, D., Grossberg, S.: How the Basal Ganglia Use Parallel Excitatory and Inhibitory Learning Pathways to Selectively Respond to Unexpected Rewarding Cues. Journal of Neuroscience 19(22), 10502–10511 (1999)
2. Faußer, S., Schwenker, F.: Learning a Strategy with Neural Approximated Temporal–Difference Methods in English Draughts. In: ICPR, pp. 2925–2928. IEEE (2010)
3. Faußer, S., Schwenker, F.: Ensemble Methods for Reinforcement Learning with Function Approximation. In: Sansone, C., Kittler, J., Roli, F. (eds.) MCS 2011. LNCS, vol. 6713, pp. 56–65. Springer, Heidelberg (2011)
4. Felleman, D.J., Van Essen, D.C.: Distributed Hierarchical Processing in the Primate Cerebral Cortex. Cerebral Cortex 1(1), 1–47 (1991)
5. Frank, M.J., Badre, D.: Mechanisms of Hierarchical Reinforcement Learning in Corticostriatal Circuits 1: Computational Analysis. Cerebral Cortex 22, 509–526 (2011)
6. Gustafsson, B., Wigström, H.: Physiological mechanisms underlying long–term potentiation. Trends in Neurosciences 11(4), 156–162 (1988)
7. Joel, D., Niv, Y., Ruppin, E.: Actor-Critic Models of the Basal Ganglia: New Anatomical and Computational Perspectives. Neural Networks 15(4-6), 535–547 (2002)
8. Malinow, R., Miller, J.P.: Postsynaptic Hyperpolarization During Conditioning Reversibly Blocks Induction of Long–Term Potentiation. Nature 320, 529–530 (1986)
9. Pennartz, C.M.A.: Reinforcement Learning by Hebbian Synapses with Adaptive Thresholds. Neuroscience 81(2), 303–319 (1997)
10. Roelfsema, P.R., van Ooyen, A.: Attention–Gated Reinforcement Learning of Internal Representations for Classification. Neural Computation 17, 2176–2214 (2005)
11. Rombouts, J.O., Bohte, S.M., Roelfsema, P.R.: Neurally Plausible Reinforcement Learning of Working Memory Tasks. In: NIPS, pp. 1880–1888 (2012)
12. Salin, P.A., Bullier, J.: Corticocortical Connections in the Visual System: Structure and Function. Physiological Reviews 75(1), 107–154 (1995)
13. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, London (1998)
14. Vitay, J., Hamker, F.H.: A Computational Model of Basal Ganglia and its Role in Memory Retrieval in Rewarded Visual Memory Tasks. Frontiers in Computational Neuroscience 4(13), 1–18 (2010)
15. Williams, R.J.: On the Use of Backpropagation in Associative Reinforcement Learning. In: ICNN, vol. 1, pp. 263–270 (1988)
16. Wörgötter, F., Porr, B.: Temporal Sequence Learning, Prediction, and Control: A Review of Different Models and Their Relation to Biological Mechanisms. Neural Computation 17(2), 245–319 (2005)

Dynamic Memory for Robot Control Using Delay-Based Coincidence Detection Neurones

Francis Jeanson and Tony White

Springer-Verlag, Computer Science Editorial,
Tiergartenstr. 17, 69121 Heidelberg, Germany
francis_jeanson@carleton.ca, arpwhite@scs.carleton.ca
http://www.springer.com/lncs

Abstract. This paper demonstrates the feasibility of dynamic memory in transmission delay coincidence detection networks. We present a low complexity, procedural algorithm for determining delay connectivity for the control of a simulated e-puck robot to solve the t-maze memory task. This work shows that dynamic memory modules need not undergo structural change during learning but that peripheral structures could be alternate candidates for this. Overall, this supports the view that delay coincidence detection networks can be effectively coupled to produce embodied adaptive behaviours.

Keywords: Dynamic Memory, Transmission Delays, Coincidence Detection, Spiking Neural Networks, Embodied Cognition.

1 Introduction

Spiking neural networks have gained significant attention in the past decade as a promising implementation mechanism for complex adaptive agent control. Despite substantial progress in the understanding of both the physiological nature of biological networks and techniques to model them formally, research remains primarily focussed on their coding capacity, structure, and dynamics as they relate to synaptic plasticity. Learning and memory, in particular, are primarily attributed to the various functional mechanisms behind connection strengthening and weakening such as hebbian learning, spike time dependent plasticity (STDP), and frequency tuning. While these are important mechanisms of neural function for adaptive behaviour, alternative mechanisms should not be ignored. Synaptic mechanisms elicit relatively slow metabolic processes (over 500ms) [12] when compared to rapid stimulus encoding which can take place in the range of 50ms [11]. The presence of active short-term mechanisms at the millisecond time scale are suggested to account for scene segmentation, rapid categorization, delayed responses, etc. [13]. In particular, mechanisms based on sustained neural dynamics via re-entrant signalling have been proposed since the 1970's [6] [15]. While a substantial amount of work has focussed on overall spike densities as potential memory signatures, few researchers have investigated their potential in realistic embodied agent tasks. Interestingly, a unique form of spatiotemporal

© Springer-Verlag Berlin Heidelberg 2013

coding referred to as coincidence detection was introduced in 1982 by Abeles [1]. Rather than behaving uniquely as frequency integrators, neurones have been shown to behave as coincidence detectors for either biophysical reasons or under specific dynamic constraints [5] [3]. A coincidence detection neurone emits a spike only if a sufficient number of incoming spikes reach it within a narrow time window, i.e., quasi-coincidentally. Contemporary interest in coincidence detection and transmission delays has led to a number of interesting findings with respect to stability criteria [2], oscillatory dynamics [8], as well as neuronal logic circuits [7]. However, little is known about distinctive dynamics that can be precisely controlled for dynamic memory function and embodied adaptive agency.

The present work introduces a method for decoding the spatiotemporal dynamics of the spiking activity into the spatial domain such that coordinated action with a sensory-motor network can enable a simulated e-puck robot to solve the t-maze task.

2 Delay Networks for Dynamic Memory

Networks of coincidence detection neurones offer distinct dynamic and informational properties from standard integrate-and-fire cells. One of their most characteristic traits comes in the form of 'firing-chains'. These are time-locked patterns of spikes that manifest as precise spatial (over cells) and temporal orders of spike events for an arbitrary input pattern. Recent work by Jeanson & White, showed that transmission delay coincidence detection networks (DCDNs) exhibit strong synchronous firing responses in narrow delay conditions while broad delays led to complex firing patterns [9]. Narrow delays were shown by Jeanson & White to be more suitable for the reactive control of a two wheeled robot in a light-seeking task. However, the broad delay regime with complex firing was hypothesized to potentially serve more complex functions. Here we replicate the simple discrete threshold model of coincidence detector spiking neurones with axonal delays from Jeanson & White [9] and determine adequate network conditions which lead to dynamic memory function. We then test this memory capacity in the embodied t-maze task by developing a novel multi-network architecture which enables robot coordination and memory driven control.

2.1 Stable Spiking

Self-sustained activity is reached reliably in these noiseless networks when single spike inputs to at least 6 neurones is made within the maximum network connection delay. By varying the connectivity in a DCD network of 25 neurones with random delays between 20 to 40 time steps, we found that a low connectivity ratio (60%) led to less predictable chaotic regimes while high connectivity (100%) led to a stable limit-cycle attractor as shown in figure 1. In the following, we make use of the limit-cycle regime to show how firing-chains can be used as dynamic memories and decoded into spatial patterns for robot control.

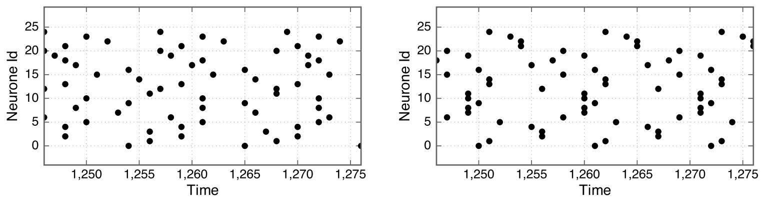


Fig. 1. Unstable spikes at 60% connectivity (left plot), yet stable at 100% (right plot)

3 T-maze Memory Task

To test the dynamic memory capabilities of DCD networks we implemented a simulated version of the T-maze memory task [14]. In this task the agent must travel autonomously up the base of the T and make a left or right turn based on the presentation of a context stimulus A or B, shortly followed by a cue stimulus X or Y. When the combined stimuli AX are presented then a left turn should be taken while a right turn is expected in the three other cases. This dual stimulus paradigm is used to show that the agent's memory network does indeed store the context stimulus for at least the time period until the cue is presented.

3.1 Sensory-Motor Network

The Enki robot simulation library was used to implement an e-puck robot with realistic noise, collision, and friction conditions [10]. A one-dimensional circular camera with a 120° field of view was divided into a left visual field (lvf) and right visual field (rvf). Visual field stimuli from either side activated a set of 5 spiking cells each. Sensory cells had refractory periods of 5 time steps (5ts) to increase sensitivity of the input while motor cells had refractory periods of 10ts. Spikes were triggered when the normalized visual fields between 0.0 and 1.0 had values of 0.35 or less. Hence we used an inverse stimulation paradigm where the absence of obstacles stimulated the agent's visual sensors. Spikes then propagated to the contra-lateral motor area which possessed 4 neurones for each wheel. These connections were fixed with many-to-many projections from each sensory cell to all motor cells in the contra-lateral sides using random delays between 10ts and 20ts. This relatively narrow delay range was chosen based on prior finding from Jeanson & White who showed that narrow delays could lead to more stable reactive behaviour [9]. Contra-lateral projections were chosen based on the successful avoidance behaviour suggested by Braitenberg vehicles [4]. Motor neurones behaved as coincidence detectors with threshold 2; hence two simultaneous spikes from sensory neurones had to reach a target motor neurone for it to fire. Finally, activation of the four respective motor neurones were summed and total activation was limited to a maximum value of 1.5 to limit maximum motor speeds. The overall sensory-motor network (SM) can be seen on the right-hand side of Figure 2.

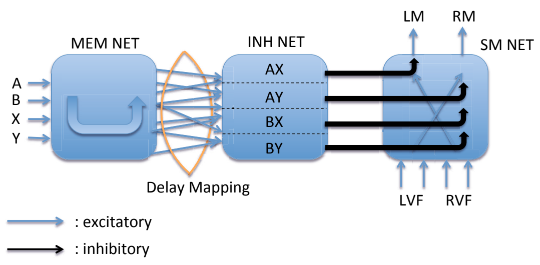


Fig. 2. Illustrated network model with Memory, Inhibitory, and Sensorimotor networks

3.2 Memory and Inhibitory Networks

The memory network (MEM) had 25 excitatory neurones which were fully connected and assigned a random uniform distribution of broad delays between 20ts and 40ts. Neurones in MEM had coincidence threshold 2 and refractory periods of 10ts. A context stimulus A or B was presented to the network after 450 time steps had gone by from the onset of a trial followed by a cue stimulus X or Y after 350 time steps. Each stimulus type consisted of synchronously activating 6 individual cells for each pattern: cells 0 – 5 for A, 6 – 11 for B, 12 – 17 for X, and 18 – 24 for Y. No other control was applied to the network during trials.

Memory network activity was projected (Proj1) from its excitatory neurones to an inhibitory network (INH) which had purpose to decode the spatiotemporal pattern in MEM and inhibit the appropriate cells in SM for correct maneuvering. INH contained 72 neurones with no internal connections but simply spiked when sufficient number of coincident signals from MEM reached a cell in INH. This coincidence detection threshold was set to 8. These in turn projected their inhibitory signal so that 1/4 of cells in INH projected to the left motor neurones in SM while the other 3/4 projected to the right motor neurones in SM. Projections from INH to SM (Proj2) were created with a fixed many-to-many scheme and had delays set randomly in the range 10ts to 20ts. Relatively narrow delays were used so that the likelihood of coincidence on the SM motor cells would be reliable enough to disrupt the stable wall avoidance behaviour and induce the appropriate turning action. The overall network is illustrated in Figure 2.

Crucially, Proj1 connections were procedurally derived with respect to delay value and connectivity based on prior runs of the memory network with each of the four combined context and cue stimulus conditions (AX, AY, BX, BY). We call this process ‘delay mapping’ which here results in Proj1 connections serving the role of transforming spatiotemporal memory patterns in MEM into spatial activation patterns in the INH network. We describe the delay mapping procedure in the next sub-section.

3.3 Delay Mapping

To obtain the delay mapping we first generated a random memory network and stimulated it with the four input conditions AX, AY, BX, and BY with fixed intervals between context and cue. Because we were only interested in the stable spatiotemporal firing pattern after the combined stimuli response had settled we chose a sufficiently late offset spike time st_o to mark the start of the region of analysis (in our case $st_o = 1400$). The limit-cycle period λ was determined from the four spike patterns ($\lambda = 11ts$). For each pattern's point process data we recorded spikes in the temporal region st_o to $st_o + \lambda$. Furthermore neurones with spike times which occurred at the same time for the same cells between lists were ignored by removing their efferent connections to INH; we call this 'collision removal'. Figure 3 shows spike times for the individual patterns and crossed spikes representing collision events between patterns. Collision removal ensured that the renewal processes were unique for the individual input conditions. We then selected an arbitrary reference time step ts_r within the period interval. Using ts_r we generated lists of spike time 'deltas' which are the time step differences of the spike time of a given cell from the reference ts_r. We denote this set of deltas for cell i of pattern ρ as Δ_i^ρ. Figure 3 illustrates the reference time step from which spike time deltas (black arrows) were calculated for pattern AY. Because at most a single spike time st_i^ρ exists for a cell i for pattern ρ during period λ we can simply compute deltas as:

$$\Delta_i^\rho = ts_r - st_i^\rho. \tag{1}$$

From these lists of deltas we then created a list of connections for each cell in MEM to INH where connections projecting to the 1/4 of cells in INH had delays determined from the list of deltas for the AX pattern. The remaining 3/4 cells in INH had connections with delays derived from the delta lists for AY, BX, and BY. A delay d_{ij^ρ} from cell i to cell j for pattern ρ is derived from the list of deltas following the equation:

$$d_{ij^\rho} = \Delta_i^\rho + d_{base}. \tag{2}$$

Where d_{base} is a base delay selected such that for refractory time R we would have $R + \lambda/2 < d_{base}$. Here we chose $d_{base} = 20$. If we compare, for example, Δ_{24}^{AY} to Δ_{23}^{AY} in Figure 3 (the two deltas at the top of the figure) we would obtain delays $d_{24j^{AY}} = 2 + 20 = 22$ for cell 24 and obtain $d_{23j^{AY}} = -2 + 20 = 18$ for cell 23. Hence all targets in INH in the region dedicated to AY would receive connections from cell 24 with delay 22ts and all connections from cell 23 with delay 18ts.

Collision removal could lead to patterns having too few distinct neurones to produce sufficient spikes for decoding by INH. However, these neurones can be 'reinstated' by assigning connections to INH with delays that do not fully coincide with an existing delay mapping. By setting delays for these connections to some arbitrary value (here 10ts), time-locked spikes in MEM could potentially coincide with spikes belonging to the the weakened memory pattern. While this process does not guarantee successful response, we observed overall improvements in correct turning behaviour.

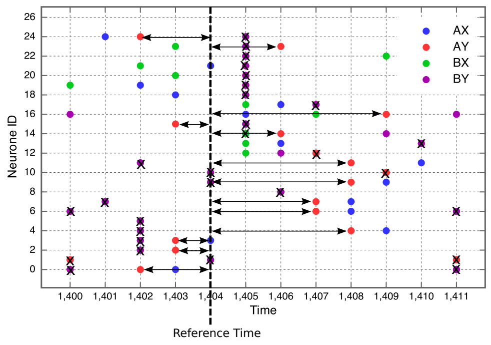


Fig. 3. Derivation of deltas in the spiking period $\lambda = 11ts$. Mapping is shown here for pattern AY (red dots only). Crossed dots are ignored collision spike times.

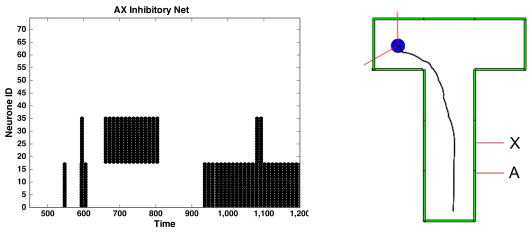


Fig. 4. Left: spike raster plot of activity in the INH network during presentation of context stimulus A and cue stimulus X. Right: robot path in the AX stimulus condition.

3.4 Results

We generated 10 random memory networks (MEM) and derived their corresponding projection connections (Proj1) to the inhibitory network. One hundred trials were tested for a given MEM/Proj1 network pair. For each run, the initial orientation of the agent was randomly set so that it would face left or right with 45° variation on either side. We also randomized the narrow delay ranges for Proj2 and SM networks on each trial.

We first describe a sample trial for stimulus condition AX. We expected that the INH network would fire such that a subset of cells detecting pattern AX should be activated after the presentation of the context stimulus A and cue stimulus X have settled. Figure 4 shows in the left raster plot the activity of the INH network during a trial with context stimulus A presented at time step 450 and cue stimulus X presented at time step 800. From the raster plot we notice that prior to time step 800 cells 18 – 35 corresponding to the recognition of pattern AY were temporally activated. This suggests that activation of memory A was sufficient to induce AY recognition. However, after the cue stimulus X was presented at 800ts and a short period of stabilization (approximately 130ts) cells 0 – 17 were activated which means memory AX was correctly decoded from MEM by the projections Proj1. This led to sufficient inhibition of the left motor neurones via projections Proj2 to the SM network. The graphic on the right hand side of Figure 4 shows the path taken by the robot when moving up the t-maze. In particular, we notice that soon after the presentation of the X stimulus a leftward turn is initiated. This led to a correct left turn reaching the top left corner of the t-maze. Similarly, correct right turns were obtained on most runs for context/cue stimuli pairs AY, BX, and BY.

We ran 100 trials for 10 random memory networks to determine the effectiveness of dynamic memory storage using the proposed delay mapping method. Between trials, random delays for Proj2 and SM connections were reset leading to slight variation for inhibition and sensorimotor control. For performance measure, we independently counted the number of correct left turns when AX was presented (25% of times) and the number of correct right turns when all three other stimuli pairs were presented (75% of times). Simulation results showed that correct left turns were performed on 82% of AX trials and correct right turns were performed on 84% of AY, BX, and BY trials leading to an overall average success rate of 83%. An agent with no memory will have equal chance of turning left or right at the end of the maze which would lead to a 50% success rate. Similarly an agent biased towards left or right turns only will also have an overall success rate of 50%. Hence agents performed the t-maze task more effectively with the dynamic memory system than without any memory.

Investigation into the individual performance of memory networks revealed that, despite collision removal and reinstating neurones, stimuli pairs could still be occasionally ambiguously decoded. This could lead to conflicting inhibition of the motor cells resulting in indecisive turning behaviour. This explained the less than perfect success rate with dynamic memory. Here we used a coincidence detection threshold of 8 for Proj1 connections for all generated memory networks. However, we believe greater memory pattern disambiguation could be achieved in future work by identifying individual decoding thresholds for each memory network that would be high enough to clearly disambiguate spatiotemporal memory patterns.

4 Conclusion

These results show that transmission delay coincidence detection neural networks can perform the role for dynamic memories whereby stimulus patterns are stored differentially and reliably without requiring internal structural/metabolic change. Importantly, this suggests that *any* random network which possesses these characteristics could potentially be exploited by adjacent networks for dynamic memory. In addition, the short settling time into periodic attractors suggests that this mechanism could subserve fast delay response integration, rapid scene categorization, and other neural functions requiring millisecond timescale exchanges. While we offer a procedural method to determine these delays, biological mechanisms from evolution, development, or other dynamical mechanisms could apply selective pressures on connections and delays leading to dynamic memories of the kind presented here. While we have not yet explored the ability for these networks to account for noise, future work will investigate the relationship between decoding thresholds and spurious neural firing.

Furthermore, accurate estimates with respect to pattern disambiguation should provide an accurate indication as to the number of patterns that can be stored in DCDNs. Overall, we hope that future computational assessments will provide additional support for the role of delay coincidence detection networks in dynamic memory and other cognitive functions.

References

1. Abeles, M.: Corticonics: Neural circuits of the cerebral cortex. Cambridge University Press (1991)
2. Arik, S.: Stability analysis of delayed neural networks. IEEE Transactions on Circuits and Systems 47, 1089–1092 (2000)
3. Bernander, O., Douglas, R., Martin, K., Koch, C.: Synaptic background activity deter- mines spatio-temporal integration in single pyramidal cells. Proceedings of the National Acedemy of Sciences 88, 1569–1573 (1991)
4. Braitenberg, V.: Vehicles: Experiments in synthetic psychology. MIT Press, Cambridge (1984)
5. Carr, C., Konishi, M.: A circuit for detection of interaural time differences in the brain stem of the barn owl. Journal of Neuroscience 10, 3227–3246 (1990)
6. Cowan, J.: Stochastic models of neuroelectric activity. In: Ricce, S., Fread, K., Light, J. (eds.) Statistical Mechanics, pp. 181–182. University of Chicago (1972)
7. Fernando, C.: Symbol manipulation and rule learning in spiking neural networks. Journal of Theoretical Biology 275, 29–41 (2011)
8. Izhikevich, E.M.: Polychronization: Computation with spikes. Neural Computation 18, 245–282 (2006)
9. Jeanson, F., White, A.: Evolving axonal delay neural networks for robot control. In: Soule, T. (ed.) Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference (GECCO 2012), pp. 121–128. ACM, New York (2012)
10. Magnenat, S., Weibel, M., Bayeler, A.: Enki: The fast 2d simulator (2007), `http://home.gna.org/enki/` (last accessed: January 28, 2013)
11. Sperling, G.: The information available in brief visual presentations. Psychological Monographs 74, 1–29 (1960)
12. Stevens, F., Wesseling, F.: Augmentation is a potentiation of the exocytotic process. Neuron 22, 139–146 (1999)
13. Thorpe, S., Imbert, M.: Biological constraints on connectionist models. In: Connectionism in Perspective, pp. 63–92. Elsevier Science Publishers (1989)
14. Ziemke, T., Thieme, M.: Neuromodulation of reactive sensorimotor mappings as a short-term memory mechanism in delayed response tasks. Adaptive Behavior 10(3), 185–199 (2002)
15. Zipser, D., Kehoe, B., Littlewort, G., Fuster, J.: A spiking network model of short-term active memory. The Journal of Neuroscience 12(8), 3406–3420 (1993)

Robust Principal Component Analysis for Brain Imaging

Petia Georgieva[1] and Fernando De la Torre[2]

[1] University of Aveiro, Aveiro, Portugal
petia@ua.pt
[2] Carnegie Mellon University, Pittsburgh, USA

Abstract. Discrimination of cognitive states from functional Magnetic Resonance Images (fMRI) is a challenging task, particularly when across subjects common representation of brain states is to be detected. Among several difficulties, the huge number of features (voxels) is a major obstacle for reliable discrimination of common patterns across brains. Principal Component Analysis (PCA) is widely applied for learning of low dimensional linear data models in image processing.The main drawback of the traditional PCA is that it is a least-square technique that fails to account for outliers. Previous attempts to make PCA robust have treated the entire image as an outlier. However, the fMRIs may contain undesirable artifacts due to errors related with the brain scanning process, alignment errors or pixels that are corrupted by noise. In this paper we propose a new dimensionality reduction approach based on Robust Principal Component Analysis (RPCA) that uses an intra-sample outlier process to account for pixel outliers. The RPCA improves classification accuracy of two cognitive brain states across various subjects compared to using conventional PCA or not performing dimensionality reduction.

Keywords: brain imaging, robust principal component analysis, functional Magnetic Resonance Imaging.

1 Introduction

Functional Magnetic Resonance Imaging (fMRI) is a powerful technique for analyzing human brain activity. Neural activity implies consumption of oxygen. The fMRI measures the level of oxygenation in the blood with respect to a control baseline. If the flow of the oxygenated blood increases this is an indicator that neural activity took place. There is a temporary increase in the ratio of oxygenated to the deoxygenated haemoglobin which affects the fMRI signal. The fMRI scanner displays a sequence of 3 dimensional images. The cells in this 3D image are called voxels (by analogy with the pixels in 2D images) and typically have a volume of a few cubic millimeters. One voxel contains hundreds of thousands of neurons.

One of the most widely used method for analysand fMRI data is Statistical Parameter Mapping (SPM) [1]. An advantage of SPM is that it produces maps

© Springer-Verlag Berlin Heidelberg 2013

describing the activation throughout the brain in response to particular stimuli. However, SPM is massively univariate, performing independent statistical tests for each voxel. Moreover SPM models and maps are strictly associated with an individual subject. Mitchell et al. [2] have advanced the discrimination of cognitive states from fMRI across subjects by developing probabilistic machine learning (ML) algorithms. They have demonstrated that a classifier trained using fMRI data from some people can successfully decode cognitive brain states of other people, indicating that our different brains encode cognitive processes in similar ways. In both approaches (SPM and probabilistic ML) the fMRI images are transformed into a time sequence of voxels (points with 3D coordinates)and the voxel patterns over time infer the brain states.

In this paper we propose an alternative approach to discriminate brain states across subjects directly from the fMRI images applying general image processing techniques. Our study is inspired by the work of Mitchell et al. [2], [3] for brain states ML classifiers and the theory of Robust Principal Component Analysis (RPCA) [4] for feature selection and dimensionality reduction. First, we apply RPCA to construct low dimensional linear-subspace representations from the noisy fMRI images and then perform standard classification. The intuition behind the RPCA approach is that fMRI based brain study is a typical case of high dimensional (huge number of voxels) but with few examples (trials) dataset, therefore some form of dimensionality reduction may improve the classification. The main candidate would be the Principal Component Analysis (PCA) that has been widely used for learning of low-dimensional linear models in image processing and computer vision. Traditional PCA constructs subspace approximation to training data that is optimal in a least- squares sense. However, the least-squares techniques fail to account for outliers. A typical form to overcome this drawback is to pre-process the training data before applying PCA. For example by manual elimination of bad images. Nevertheless, when automated learning is applied to more realistic problems, and the amount of training data increases, it becomes impractical to manually verify that all the data is ”good”. Previous attempts to make PCA robust have treated the entire image as an outlier. However, the fMRIs may contain undesirable artifacts due to errors related with the brain scanning process, alignment errors or pixels that are corrupted by noise. It involves intra-sample outliers which effect some but not all of the pixels in the image. In [4] a theory of Robust PCA (RPCA) is developed that uses an intra-sample outlier process to account for pixel outliers.

In our previous work ([5]), RPCA was applied to extract features from fMRI data associated with each subject and then individual Gaussian Naive Bayes(GNB) classifiers were trained to discriminate among two cognitive brain states. For each test case, the probability of the correct class was higher than the probability of the incorrect class, thus a perfect binary classification was obtained. In the present paper we consider the more general problem of cognitive states discrimination across subjects. Based on integrated fMRI data from multiple subjects, we want to develop across subjects supervised learning of common representations and prove that different brains encode cognitive processes in a similar way. A feedforward Artificial Neural Network (ANN) with probabilistic activation functions was applied as a classifier. The paper proceeds as follows. Section 2 introduces RPCA formally, section 3 provides results on the application of the RPCA-ANN to empirical fMRI dataset, and finally in section 4 a discussion of the results is provided.

2 Robust Principal Component Analysis

PCA is a statistical technique that is useful for dimensionality reduction. Let $\mathbf{D} = [\mathbf{d}_1\mathbf{d}_2...\mathbf{d}_n]$ be a matrix $\mathbf{D} \in \mathbb{R}^{d\times n}$, where each column $\mathbf{d}_i = [d_i^1 d_i^2 ... d_i^d]^T$ is an image, n is the number of training images, and d is the number of pixels in each image. We assume that training data is zero mean, otherwise the mean of the entire data set is subtracted from each column $\mathbf{d}_i$. Let the first k principal components of $\mathbf{D}$ be $\mathbf{B} = [\mathbf{b}_1\mathbf{b}_2...\mathbf{b}_k] \in \Re^{d\times k}$. The column of $\mathbf{B}$ are the directions of the maximum variation within the data. The principal components maximize $max_{\mathbf{B}} \sum_{i=1}^{n} \|\mathbf{B}^T\mathbf{d}_i\|_2^2 = \mathbf{B}^T \Gamma \mathbf{B}$ with the constraint $\mathbf{B}^T\mathbf{B} = \mathbf{I}$, where $\Gamma = \mathbf{D}\mathbf{D}^T = \sum_i \mathbf{d}_i\mathbf{d}_i^T$ is the covariance matrix. The column of $\mathbf{B}$ form a basis that spans the principal subspace. If the effective rank of $\mathbf{D}$ is much less than d, we can approximate the column space of $\mathbf{D}$ with $k << d$ principal components. The data $\mathbf{d}_i$ can be approximated by the linear combination of the principal components as $\mathbf{d}_i^{rec} = \mathbf{B}\mathbf{B}^T\mathbf{d}_i$ where $\mathbf{B}^T\mathbf{d}_i = \mathbf{c}_i$ are the linear coefficients obtained by projecting the training data onto the principal space; that is $\mathbf{C} = [\mathbf{c}_1\mathbf{c}_2...\mathbf{c}_n] = \mathbf{B}^T\mathbf{D}$. In [4] the RPCA is formulated as minimization of the following error function

$$E_{rpca}(\mathbf{B}, \mathbf{C}, \mu, \sigma) = \sum_{i=1}^{n} e_{rpca}(\mathbf{e}_i). \tag{1}$$

where, $e_{rpca} = \mathbf{e}_i^T\mathbf{e}_i$ is the energy of the reconstruction error $\mathbf{e}_i(\mathbf{d}_i - \mu - \mathbf{B}\mathbf{c}_i, \sigma)$, with mean $(\mathbf{d}_i - \mu - \mathbf{B}\mathbf{c}_i)$ and variance $\sigma = [\sigma_1\sigma_2...\sigma_d]$ that specifies a scale parameter for each of the d pixel locations. In standard PCA, the number of bases is usually selected to preserve the ratio between the energy of the reconstructed vectors and the original ones larger than some percentage. In RPCA this criterion is not straightforward to apply. The robust error e_{rpca} depends also on σ. Therefore we first apply standard PCA to the data, and calculate the number of bases which preserve the ratio between the energy of the reconstructed vectors and the original ones larger than 0.55; With this initial number of bases, we apply RPCA, minimizing (1), until convergence. At the end of this process we have a matrix $\mathbf{W}$ that contains the weights of each pixel in the training data. We detect outliers with this matrix and set values of $\mathbf{W}$ to 0 if w_{pi} is above a certain threshold and to w_{pi} otherwise, obtaining $\mathbf{W}^*$. We then incrementally add additional basis and minimize $E(\mathbf{B}, \mathbf{C}, \mu) = \|\mathbf{W}^* \circ (\mathbf{D} - \mu - \mathbf{B}\mathbf{C}\|_2^2$ but maintaining constant weights $\mathbf{W}^*$. The iterative minimization of (1)starts with an initial guess for $\mathbf{B}$ chosen to be the mean of D plus random Gaussian noise. The RPCA formulation has an inherent scale parameter that determines what is considered an outlier.

3 Experiments and Results

The RPCA-ANN approach is demonstrated on the Star-Plus (SP) benchmark fMRI data known from other studies [6], [2]. We used the SP dataset available for public use [7] where six participants were presented with a sequence of 40 trials. In half of the trials participants were shown first a picture (4 sec.) followed by a blank screen (4 sec.), and then a sentence (4 sec.). The participants had to press a button indicating whether the sentence correctly described the picture. Then they rested before the next trial began. In the other half of the trials the sentence was presented first and the picture second, using the same timing. Images were collected every 500msec. Only a fraction of the brain of each subject was imaged. The data is marked up with 25-30 anatomically defined regions called "Regions of Interest" (ROI).

The goal of the study was to train a classifier to distinguish whether the subjects are viewing a picture or sentence. Therefore two brain states are considered: state P related with the stimulus Picture that represents visually evoked brain activity and state S associated with the stimulus Sentence and corresponds to decision making brain state. In the rest of this paper, we will denote the data related with each state as P dataset and S dataset respectively. For each subject we have 80 trials in total (20 P1, 20 P2, 20 S1 and 20 S2 trials). The numbers associated with the trials indicate if the picture/sentense was the first (P1/S1) or the second (P2/S2) stimulus. Half of the trials of all six subjects were used as training data and the rest of the trials of the subjects for testing.

For each trial of fMRI data we have a sequence of 16 volume (3 dimensional) images, one volume consists of 8 2D snapshots. The data matrix D has 128 columns (16×8) that corresponds to the total number of 2D images and (64×64) lines corresponding to the number of pixels in each image. Before applying the RPCA, a test for data variability was performed by the Singular Values (SV) of matrix D. The first nine SVs carry more than 90% of the total signal energy (see Fig.1), therefore the number of bases (the principal components) was set to $k = 9$. RPCA is applied for the fMRI images of all trials. If the optimization procedure to compute the projection matrix is executed for each trial (unsupervised mode) the computational time would be rather high. Therefore we apply the so called supervised mode where the optimal projection matrix is computed ones (for one trial) and applied for the rest of the trials. The effect of applying RPCA for image samples from four trials of one of the subjects is illustrated on Fig.2. The reconstructed images are the inputs of the classifier. Our goal is to train a classifier with fMRI data from $n-1$ subjects to automatically decode the cognitive states of the nth subject. The ANN learns a class-conditional distribution model for each feature (each pixel of the training images) over the 40 training trials from $(n-1)$ subjects given the class label (supervised learning). The ANN assigns the class for the test trials from the nth subject applying the Maximum Likelihood Estimation (MLE). The procedure is repeated for each subject. On Fig. 3, Fig. 4 and Fig. 5 are depicted the MLE probabilities related with the discriminated cognitive brain states (P and S) of all subjects in the fMRI dataset over all (40) test trials. The first 20 trials on the figures, correspond to the S state and the next 20 to the P state. Note that the proposed

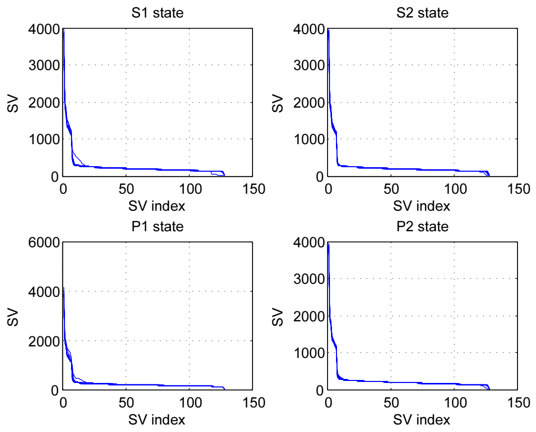


Fig. 1. Singular Values (SV) of fMRI image matrix (one trial)

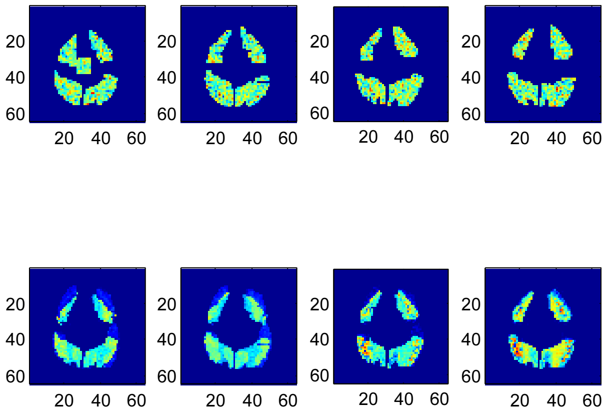


Fig. 2. Subject 1: *Top:* Original images. *Bottom:* RPCA reconstructed images

classification technique RPCA-ANN assigns the correct class for almost all test trials (Fig. 3). while two alternative approaches ANN (Fig. 4) and PCA-ANN (Fig. 5) demonstrate poor generalization properties.

The performance of the RPCA-ANN, ANN and PCA-ANN in terms of % of correctly classified test trials for each of the 6 subjects is summarized on Table 1. The ANN attempts to directly classify the original fMRI data. The PCA-ANN technique represents each image as a linear combination of k principal components. However, pixel outliers in the images are not eliminated (see the top images on Fig. 2) and their presence in the principal components degrade the classification. The proposed RPCA-ANN technique first applies PCA, then accounts for the pixel outliers and eliminate them by substituting the value of W associated with that pixel with zero. The elimination of the pixel outliers additionally reduce the feature dimension of each image and thus leads to a better classification of the subject's test trials.

Table 1. % of correctly classified test trials

Subject	ANN	PCA-ANN	RPCA-ANN
1	69	73	85
2	35	35	87
3	59	60	83
4	57	57	82
5	43	45	89
6	56	57	85

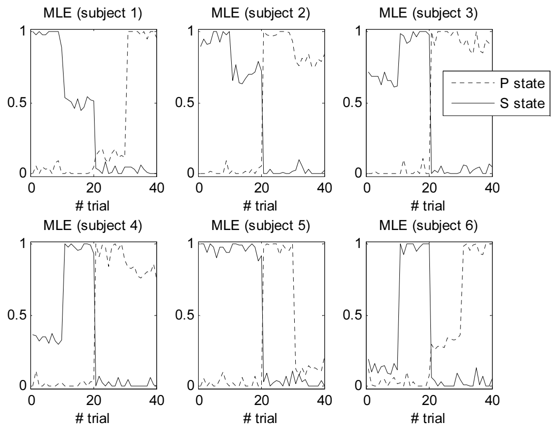


Fig. 3. Maximum Likelihood Estimation (MLE) with RPCA-ANN (test data)

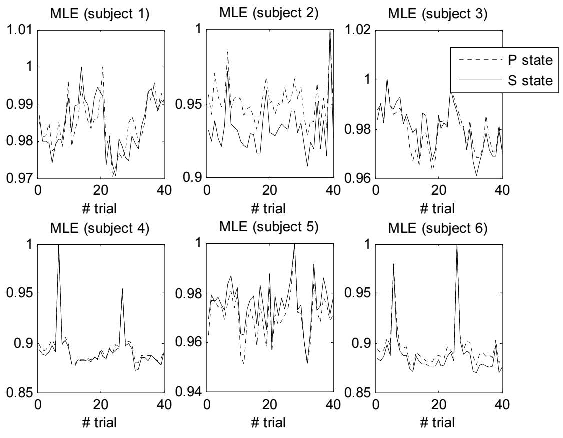


Fig. 4. Maximum Likelihood Estimation (MLE) with ANN (test data)

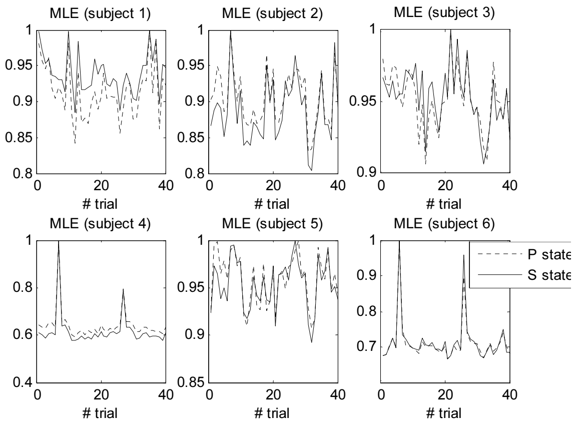


Fig. 5. Maximum Likelihood Estimation (MLE) with PCA-ANN (test data)

4 Conclusions

In this paper is presented a method for robust PCA that improves the discrimination of cognitive brain states from fMRI across subjects. The method is illustrated on empirical data and shows that it improves classification accuracy relative to using conventional PCA or not performing dimensionality reduction. The results reported here are obtained with the combination of RPCA and an ANN classifier. However, other classifiers like Gaussian Naive Bayes (GNB) and Support Vector Machine (SVM) were also tested with similar conclusions. We are aware that the hard binary decision where one out of two states is selected could not be realistic scenario in most of the cases. However our goal is not to exhaustively search for a technique to decode as many cognitive brain states as possible. Instead, we want to clearly illustrate that brain states discrimination across subjects can be significantly improved by RPCA.

Acknowledgements. This work was funded by the Portuguese Foundation for Science and Technology (FCT) in the framework of the CMU-Portugal Faculty Exchange program 2012 and partially funded by FEDER through the Operational Program Competitiveness Factors - COMPETE and by National Funds through FCT in the context of the projects FCOMP-01-0124-FEDER-022682 (FCT reference PEst-C/EEI/UI0127/2011) and Incentivo/EEI/UI0127/2013.

References

1. Friston, K.J.: Introduction to statistical parametric mapping. In: Frackowiak, et al (ed.) Human Brain Function (2003)
2. Mitchell, T., Hutchinson, R., Niculescu, R., Pereira, F., Wang, X.: Learning to Decode Cognitive States from Brain Images. Machine Learning 57, 145–175 (2004)

3. Hutchinson, R., Niculescu, R., Keller, T., Rustandi, I., Mitchell, T.: Modeling fMRI data generated by overlapping cognitive processes with unknown onsets using Hidden Process Models. NeuroImage 46, 87–104 (2009)
4. De la Torre, F., Black, M.: Robust Principal Component Analysis for Computer Vision. In: IEEE Int. Conf. on Computer Vision, Vancouver, Canada (2001)
5. Georgieva, P., Nuntal, N., De la Torre, F.: Robust Principal Component Analysis for improving cognitive brain states discrimination from fMRI IbPRIA. In: 6th Iberian Conference on Pattern Recognition and Image Analysis Madeira, Portugal, June 5-7 (accepted, 2013)
6. Keller, T., Just, M., Stenger, V.: Reading span and the time-course of cortical activation in sentencepicture verification. In: Annual Convention of the Psychonomic Society, Orlando, FL. (2001)
7. `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-81/www/`

Phase Control of Coupled Neuron Oscillators

Mayumi Irifune and Robert H. Fujii

Computer Systems Department, University of Aizu, Aizu Wakamatsu City, Fukushima Prefecture, Japan
fujii@u-aizu.ac.jp

Abstract. The phase response of an Izhikevich neuron integrator/resonator model based oscillator to a weak short-duration external input pulse is used to determine the Izhikevich model dynamic parameter values needed to attain a specified phase difference between coupled neuron oscillators working at the same natural oscillation frequency. The design of a new type of neuron oscillator-chain based artificial central pattern generator for the coordinated four-legged animal walking movement is proposed as an application.

Keywords: Izhikevich neuron model, saddle-node bifurcation, Andronov-Hopf bifurcation, integrator, resonator, stable limit cycle, phase difference, phase response curve, weakly-coupled oscillators.

1 Introduction

Various synchronization phenomena exist in the biological world. For example, the repetitive muscle movements in swimming and cardiac rhythmic movements [2] are believed to result from a system composed of coupled oscillating neurons that act in synchrony while maintaining various phase differences with respect to each other.

In this paper, the Izhikevich neuron model of a cortical neuron [5,6,7,8] is used to model an integrator as well as a resonator type neuron acting with oscillatory behavior. The Izhikevich neuron model has its roots in the Hodgkin-Huxley (HH) neuron model [9] but it is a considerably simpler model; it approximates the action potential function of the HH neuron model and captures the essence of the $I_{Na}+ I_K$ ionic current dynamics that allow 2D dynamical system phase plane analysis and numerical simulations of the neuron action potential as shown in Figs. 1(a) and (b) respectively. The Izhikevich cortical neuron model [7] can be described by the following equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

$$if \quad v > v_{peak}, \quad then \quad \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases}.$$

© Springer-Verlag Berlin Heidelberg 2013

The variable v represents the action potential. The variables u represents the action potential recovery variable that takes into account ionic currents, such as the activation of K ionic current and the inactivation Na ionic current. The parameter 'a' describes a time scale value for the recovery variable u. The parameter 'b' describes a sensitivity factor value for the recovery variable u. The parameter 'c' describes the reset voltage value for the action potential v. The parameter 'd' describes the reset value of the recovery variable u after an output spike has occurred. The neuron phase portrait (solid curve with arrow followed by dark dashed line) in the vu phase plane is shown in Fig. 1(a). The neuron action potential as a function of time is shown in Fig. 1(b). Since the Izhikevich model can generate the upstroke action potential (v^2) but not the downstroke action potential, the Izhikevich model resets the membrane potential when the membrane potential v exceeds v_{peak}.

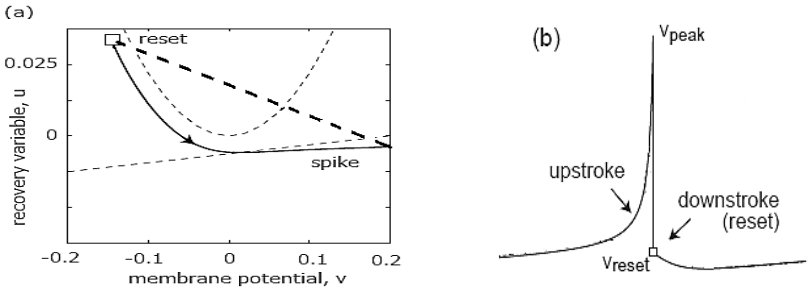


Fig. 1. Neuron behavior: (a) vu plane and (b) action potential (v) vs. time

The resting state is equivalent to the stable equilibrium point in the dynamical system of the neuron. A periodic spiking state (i.e. oscillatory output) corresponds to the neuron limit cycle dynamics. When a neuron model parameter value such as current I is changed, the neuron state can shift states. When a smooth small change in neuron parameter value(s) causes the neuron behavior to suddenly change, a bifurcation is said to have occurred. In this paper, the oscillatory behavior of neurons that undergo saddle-node (integrator type) bifurcations [7] and Andronov-Hopf bifurcations (resonator type) [7] are considered.

Both the integrator and resonator model neurons can be made to oscillate with some specified natural frequency. A phase response curve or phase resetting curve (PRC) [1,3] represents the amount of phase shift that occurs when a weak (i.e. small magnitude) [8] input pulse stimulus of short duration (relative to the period of oscillation) is applied at the input of the oscillating neuron. If the phase resetting curve value is positive (negative), the phase is advanced (delayed). An example PRC for a saddle-node bifurcation neuron model with some constant output spiking oscillation is shown in Fig. 2(a) where the y-axis represents the amount of phase shift and the x-axis represents the time relative to the oscillator period (period = 40 ms. in the figure) when the input stimulus pulse was received. A PRC can be defined as follows:

$$PRC(\theta) = \frac{T_n - T_{new}}{T} for\ n \in \mathbb{N}. \tag{3}$$

where the phase $\theta = t/T$ with t denoting the time since the last output spike of the oscillator and T denoting the oscillator neuron period. T_n denotes the time of nth spike when no external stimulus is present (dotted line in Fig. 2(b)), and T_{new} denotes the time of the next spike after the external stimulus was received (solid line in Fig. 2 (b)). The overall phase shift over many periods is composed of the current phase of the oscillator (relative to some absolute reference) and the phase shift due to the most recent external stimulus pulse.

$$\theta_{new} = \theta_{current} + PRC(\theta_{current}) \qquad mod\,T. \tag{4}$$

$\theta_{current}$ describes the phase of the oscillator at the instant when the input stimulus pulse is received. *Mod T* stands for modulo T. T is the oscillator frequency. The Izhikevich cortical neuron model Equations (1) and (2) were used to obtain the Phase response Curve (PRC) shown in Fig. 2 (a).

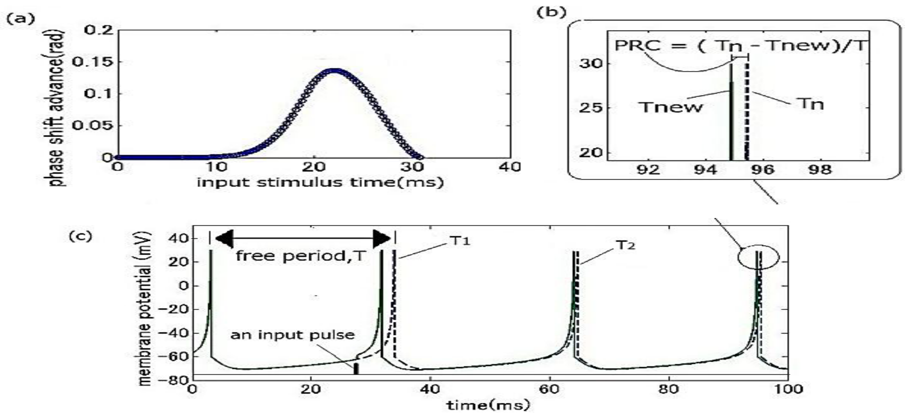


Fig. 2. (a) Phase response curve for Izhikevich neuron model with a = 0.01, b = 0, c = - 60, d = 6, I = 35pA, and external input stimulus pulse strength of 0.05pA. (b) Phase shift due to external stimulus. (c) Neuron membrane potential plot showing phase shift due to external stimulus.

The goal of this research was to analyze how neuron oscillators could be made to have absolute/traveling-wave phase differences between them. In order to do so, the behavior of weakly coupled [8] integrator/resonator neuron oscillators operating at identical natural frequencies was analyzed using PRCs under the assumption that each oscillator had a stable limit cycle and that their coupling strengths were small.

2 Phase-Shift Analysis of Single Neuron Oscillators

Izhikevich model parameters a, b, c, d and the current I were selected appropriately so that both the saddle-node and sub-critical Andronov-Hopf bifurcation neurons could be made to oscillate.

Two example PRCs for an Izhikevich neuron undergoing sub-critical Andronov-Hopf (AH) bifurcations while receiving excitatory external weak short stimulus pulses are shown in Fig. 3. It should be noted that the AH PRCs have both positive and negative phase shift values. The X-axis range is 0 - 2 π radians.

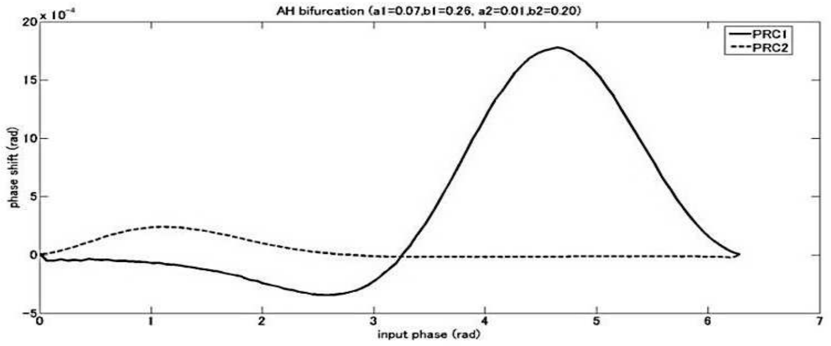


Fig. 3. PRC1 has a = 0.07, b = 0.26 and PRC2 has a = 0.01, b = 0.20

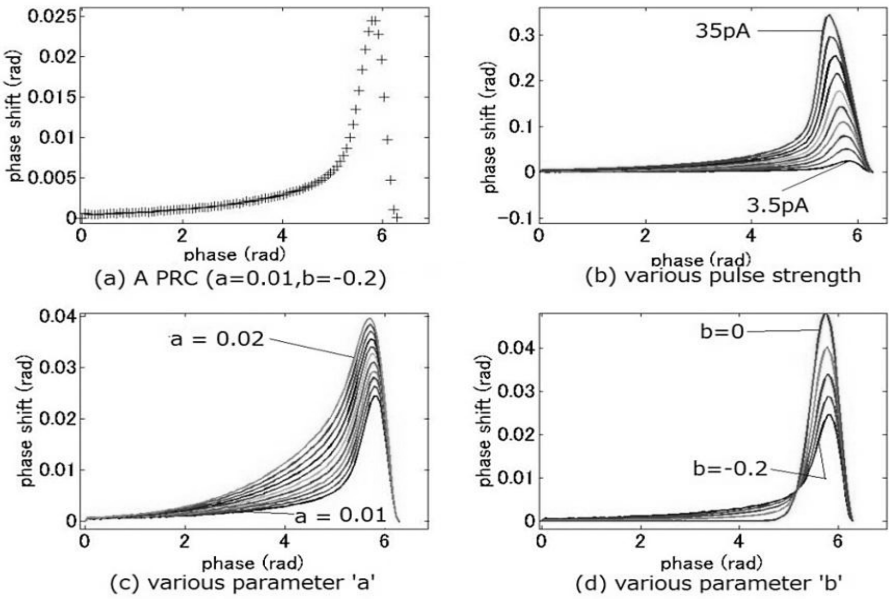


Fig. 4. (a) a = 0.01, b= -0.2, c = -60, d = 6, and I = 17pA. (b) Input stimulus pulse strengths from 3.5pA to 37.5pA in increments of 3.5pA applied. (c) Parameter 'a' varied from 0.01 to 0.02 in increments of 0.001. (d) Parameter 'b' varied from -0.2 to 0 in increments of 0.05.

PRCs for the Izhikevich neuron undergoing saddle-node bifurcations are shown in Fig. 4; when excitatory (inhibitory) stimulus pulses occur, the PRCs have only positive (negative) values [3]. As can be seen in Figs. 4 (b), (c), and (d), increasing parameters 'a', 'b', and/or external input stimulus pulse magnitude made the peak of the PRC larger and shifted it to the left. The PRC y-axis represents the phase shift in radians and the PRC x-axis represents the phase of the oscillator at the time the external weak short pulse stimulus is received. The x-axis range is 0 to 2 π radians.

3 Phase Analysis of Two-Coupled Oscillators

The behavior of two oscillating neurons whose outputs are mutually coupled to each other can be described using phase shift equations [10,11].

$$\frac{d\theta_1}{dt} = \Omega_1 + \epsilon_1 H_{12}(\theta_1 - \theta_2) \ . \tag{5}$$

$$\frac{d\theta_2}{dt} = \Omega_2 + \epsilon_2 H_{21}(\theta_2 - \theta_1) \ . \tag{6}$$

In the above equations, i = 1 or 2 for the two neuron sytem, Ω_i = natural oscillator i frequency, ε_1= coupling strength of output of oscillator 2 to oscillator 1, ε_2 = coupling strength of output of oscillator 1 to oscillator 2, H_{ij} = shift in phase of oscillator i due to input from oscillator j, $\theta_1 - \theta_2$ = phase of oscillator 1 when oscillator 2 fires a pulse, $\theta_2 - \theta_1$ = phase of oscillator 2 when oscillator 1 fires a pulse.

The following describes the phase difference between the two oscillators:

$$\frac{dX}{dt} = w + G(X) \, . \tag{7}$$

$X = \theta_1 - \theta_2$, $w = \Omega_1 - \Omega_2$, $G(X) = \varepsilon_1 H_{12}(X) - \varepsilon_2 H_{21}(-X)$. Thus, G(X) is the graph obtained as the difference between the PRC1 graph and the mirrored PRC2 graph. Since the two oscillators have the same frequency, w = 0. A constant phase difference between the two oscillators is achieved when dG(X)/dX is negative (stable equilibrium) and dX/dt = G(X) = 0.

Oscillators that have different PRCs can be selected to achieve a desired constant phase difference. Whether the phase difference is a constant absolute phase difference or a constant traveling-wave type phase difference depends on the PRC values where G(X) = 0 occurs. If the PRC values are 0 at G(X) = 0, a constant absolute phase difference can be achieved otherwise, a traveling-wave type phase difference is achieved. For sub-critical Andronov-Hopf (AH) bifurcation oscillators, the G(X) = 0 point usually corresponds to a point where the PRCs are not 0, thus a constant but traveling wave-type phase difference between the two oscillators is achieved. Two different G(X) graphs for sub-critical AH bifurcation oscillators are shown in Fig. 5.

Achievable phase difference ranges for a pair of coupled saddle-node/AH bifurcation oscillators are shown in Tables 1 and 2. Phase difference ranges are larger in Table 2 than in Table 1 because in Table 2 the a, b parameter values and the ε values are varied instead of just varying the coupling strength ε value.

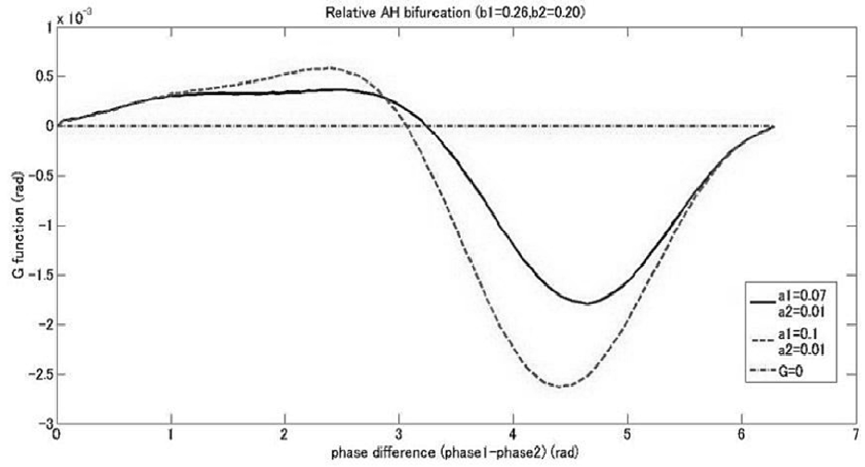


Fig. 5. Sub-critical AH oscillators G(X) graphs. When PRCs ≠0 at G(X) = 0 and dG(X)/dX= negative, the constant phase difference is a traveling wave.

Table 1. Examples of achievable phase difference ranges for a pair of mutually coupled saddle-node and AH bifurcation neuron oscillators. The coupled oscillators have the same a and b values, but same/different coupling strength ε values.

	ε_1 = -1; ε_2 = -1 to -3	**ε_1 = -1 to -3; ε_2 = -1**
Both SN oscillators a = 0.01, b = - 0.2	3.14 to 4.21 radians	2.07 to 3.14 radians
Both AH oscillators a = 0.1, b= 0.26	3.14 to 3.178 radians	3.105 to 3.14 radians

Table 2. Achievable phase difference for a pair of coupled sub-critical AH bifurcation oscillators. The coupled oscillators have various a, b, and ε values.

a_1=0.01, a_2=0.01 to 0.1	**ε_1=ε_2 = -0.1**	**ε_1= -1, ε_2=-0.3**
b_1=0.26, b_2= 0.05 to 0.26	2.7453 to 3.8401 radians	2.6742 to 3.9135 radians
b_1=0.20, b_2= 0.05 to 0.26	2.6885 to 3.8231 radians	2.6681 to 3.9195 radians
b_1=0.15, b_2= 0.05 to 0.26	2.6903 to 3.8058 radians	2.6637 to 3.9022 radians
b_1=0.10, b_2= 0.05 to 0.26	2.6553 to 3.7892 radians	2.6529 to 3.8799 radians
b_1=0.05, b_2= 0.05 to 0.26	2.4430 to 3.7751 radians	2.5312 to 3.8754 radians

4 Chain of Oscillators

With a knowledge of the range of possible phase differences achievable with two-coupled oscillators, it is possible develop a chain of two-coupled oscillators that can be used to design an artificial central pattern generator [10,11] that mimics the synchronized and phase shifted walking leg movements of a four-legged animal. A new type of oscillator chain that can achieve such movements is shown in Fig. 6.

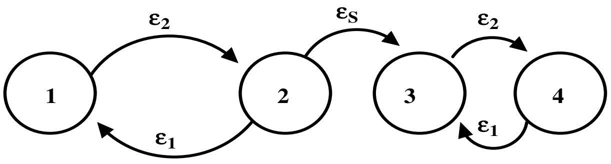


Fig. 6. Chained Oscillator System (COS) comprised of coupled oscillators 1 and 2 chained to coupled oscillators 3 and 4

It is assumed that all oscillator neurons 1 - 4 are AH bifurcation oscillators. A traveling-wave type constant phase shift between coupled oscillators 1 and 2 as well as between coupled oscillators 3 and 4 can each be set to 3.883 rad. by a proper selection of inhibitory (negative) ε_i (i= 1 or 2) values, and a_i and b_i parameter values (see Table 2). The chaining of the coupled oscillators 1-2 to 3-4 is accomplished with a positive ε_S (i.e. excitatory) coupling strength. The phase shift between oscillators 2 and 3 will be approximately 0 radians because the PRC for oscillator 3 has a negative slope (i.e. a stable point) near 0 and 2π radians. The chaining between oscillators 2 and 3 is needed for synchronization.

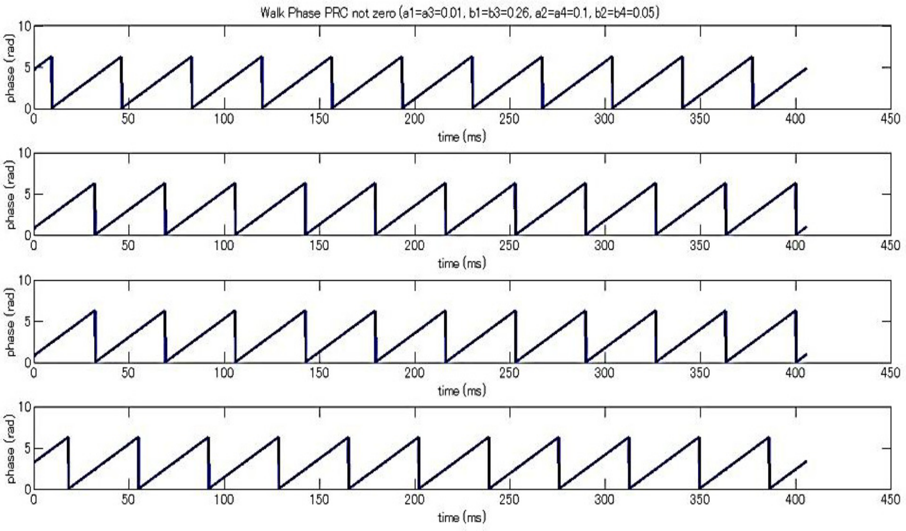


Fig. 7. Outputs for chain of oscillators shown in Fig. 6. Top output is for oscillator 1 followed by oscillator 2, oscillator 3, and bottom output for oscillator 4.

Thus, the constant phase difference (either absolute or a traveling-wave-type) between oscillator 1 and oscillator 4 will be 1.5 radians (i.e. (3.883 radians + 3.883

radians) mod. 2π) $\simeq$1.5 radians $\simeq \pi/2$). If the COS shown in Figure 6 is chained to another COS of identical specification, another phase shift of approximately $\pi/2$ radians can be achieved. Hence, to achieve the synchronized phase shifted walking leg movements of a four-legged animal with each leg moving $\pi/2$ apart in time, three COSs (12 neurons) will be needed. The phase shifted traveling wave-type of outputs for the COS oscillators 1 (top) through 4 (bottom) in Fig. 6 are shown in Fig. 7. If oscillator 3 was a SN bifurcation oscillator instead of an AH bifurcation oscillator, it would not be possible to have a 0 phase shift difference between oscillators 2 and 3.

5 Conclusions

Phase response curves (PRCs) of Izhikevich neuron model based oscillators were obtained through simulations. Using this database of PRCs, it was possible to determine the range of constant absolute or traveling-wave type phase difference that could be achieved between two mutually coupled oscillators. A manual search of the appropriate neuron parameter values was carried out in order to apply the proposed method for the coordinated walking movement of a four-legged animal. An automated computer search and optimization algorithm that uses simulated PRC data or numerically generated Malkin's theorem [4,8] based PRC data can be developed.

References

1. Ermentraut, B.: Type I Membrane, Phase Resetting Curves and Synchrony. Neural Computation 8, 979–1001 (1966)
2. Brodfuehrer, P.D., Debski, E.A., O'Gara, B.A., Friesen, W.O.: Neuronal Control of Leech Swimming. J. of Neurobiology 27(3), 403–418 (1995)
3. Gutkin, B.S., Ermentrout, G.B., Reyes, A.D.: Phase-response Curves Give the Responses of Neurons to Transient Input. J. of Neurophysiology 94, 1623–1635 (2005)
4. Malkin, I.J.: Some Problems in the Theory of Nonlinear Oscillations. Moscow (1956); U.S. Atomic Energy Commission, translation AEC-tr-3766, Washington D.C (1959)
5. Izhikevich, E.M.: Simple Model of Spiking Neurons. IEEE Trans. on Neural Networks 14, 1569–1572 (2003)
6. Izhikevich, E.M.: Which Model to Use for Cortical Spiking Neurons? IEEE Trans. on Neural Networks 15, 1063–1070 (2004)
7. Izhikevich, E.M.: Dynamical System in Neuroscience: The Geometry of Excitability and Bursting. The MIT Press, Massachusetts (2007)
8. Hoppensteadt, F.C., Izhikevich, E.M.: Weakly Connected Neural Networks. Springer, New York (1997)
9. Hodgkin, A.L., Huxley, A.F.: A Quantitative Description of Membrane Current and Application to Conduction and Excitation, Nerve. J. of Physiology 117, 550–554 (1952)
10. Kopell, N.: Toward a Theory of Modeling Central Pattern Generators. In: Cohen, A., Grillner, S., Rossignol, S. (eds.) Neural Control of Rhythmic Movements in Vertebrates, pp. 369–413. J. Wiley & Sons, New York (1988)
11. Rand, R.H., Cohen, A.H., Holmes, P.J.: System of Coupled Oscillators as Models of Central Pattern Generators. In: Cohen, A., Grillner, S., Rossignol, S. (eds.) Neural Control of Rhythmic Movements in Vertebrates, pp. 333–367. J. Wiley & Sons, New York (1988)

Dendritic Computations in a Rall Model with Strong Distal Stimulation

Youwei Zheng and Lars Schwabe

Faculty of Computer Science and Electrical Engineering
Adaptive and Regenerative Software Systems
University of Rostock

Abstract. Rall's work is the basis for investigating dendritic computations, but only recently the technology became available to study their properties experimentally. Empirical evidence supports the idea that synaptic inputs at *distal* dendritic locations set the context for recognizing synaptic activation patterns of synapses *proximal* to the soma. Such a context-dependence is fundamental for action selection and decision making. It is usually assumed that *active* channels in dendrites are necessary. Here we investigate under which conditions of synaptic drive, a *passive* dendrite model can realize such a context-dependence, and we find that stronger distal than proximal activation, paired with delayed inhibition, is sufficient to produce so-called up states. Testing the model on a different protocol (selectivity to synaptic activation sequences: distal to proximal vs. proximal to distal) shows that it is more similar to recent experimental findings than Rall's original parameterization, and similar to a model with active dendrites. Our results show that, given stronger distal activation, context-dependent pattern recognition can be implemented in passive dendrites. As a consequence, future experimental studies need to determine on a case-by-case basis the contribution of active channels in dendrites (a single neuron property) vs. synaptic drive (a network property) in context-dependent pattern recognition.

Keywords: Computational neuroscience, dendrites, pattern recognition, classification.

1 Introduction

The pioneering exploration of the computational capabilities of dendrites by Rall's legacy has provided the theoretical framework for computational neuroscience. Many of his works are classics, from modeling the complex branching structures of dendritic trees to studying their contributions to the signal processing in the nervous system. One landmark-paper is the *Theoretical significance of dendritic trees for neuronal input-output relations* [4] published in 1964. It was the first theoretical paper to demonstrate neurons as directionally sensitive computational units, which are able to detect the spatiotemporal direction of synaptic activation on the dendrites.

© Springer-Verlag Berlin Heidelberg 2013

Rall showed that the farther the location of stimulation is away from the soma, the lower and later the voltage peaks are observed at the soma. However, this prediction is not compatible with our current knowledge on dendritic location-independence of unitary somatic EPSP [2]. Moreover, a recent work on synaptically driven state transitions of striatal spiny neurons revealed a similar property and discovered that the somatic up state can be attributed to the activation of a group of adjacent distal spines in a rapid succession [3]. Interestingly, despite distinct up state durations, the peak somatic voltages do not show location-dependent differences.

One functional role of such synaptically driven state transitions may be in performing context-dependent pattern recognition, i. e. only in the up state patterns are recognized while in the down state the neuron as a classifier is switched off. In this paper, motivated by Rall's work [4], which demonstrated rich computational capabilities of dendrites such as the direction selectivity without assuming voltage-dependent channels, we explore conditions under which a Rall model predicts up states and direction sensitivity. All simulations were done on exactly the model neuron used by Rall. Fig. 1 shows Rall's model and our reproduction of one of Rall's results in [4], namely the somatic voltage response when pairs of compartments at different distances from the soma receive excitatory stimulation. In the rest of the paper we only change the stimulation of this model, but not the model parameters.

We show that Rall's model captures the key characteristics of up state transitions triggered by the activation of distal dendrites such as in striatal spiny neurons [3]. As expected from a Rall model, we predict that distal excitatory sites need to be stimulated more strongly than proximal ones to achieve a location independence of the somatic EPSP [5]. However, to be consistent with the experimentally reported voltage decay from the up state we find balanced and delayed inhibition to be necessary. We also show that Rall's model, when stimulated with the location-dependent excitatory inputs we applied, predicts direction-selective responses closer to recent experimentally reported direction-selectivity [1] than Rall's original protocol [4]. Finally, we show that the location-dependent strength of excitatory inputs leads to model predictions, where strong distal *and* proximal activation are required to depolarize the soma close to threshold; proximal activation alone, or proximal activation paired with weak distal activations were not sufficient.

2 Model Description

We used exactly the same model as in Rall's original work. A full model description is omitted for the sake of brevity, however, for completeness we summarize the main features of a Rall model as follows: 1) the dynamics of the membrane potential V_m are calculated using an equivalent electrical circuit model. The basic form of such a model is

$$C_m \frac{dV_m}{dt} = -G_r(V_m - E_r) - G_e(E_e - E_r) - G_i(E_i - E_r), \qquad (1)$$

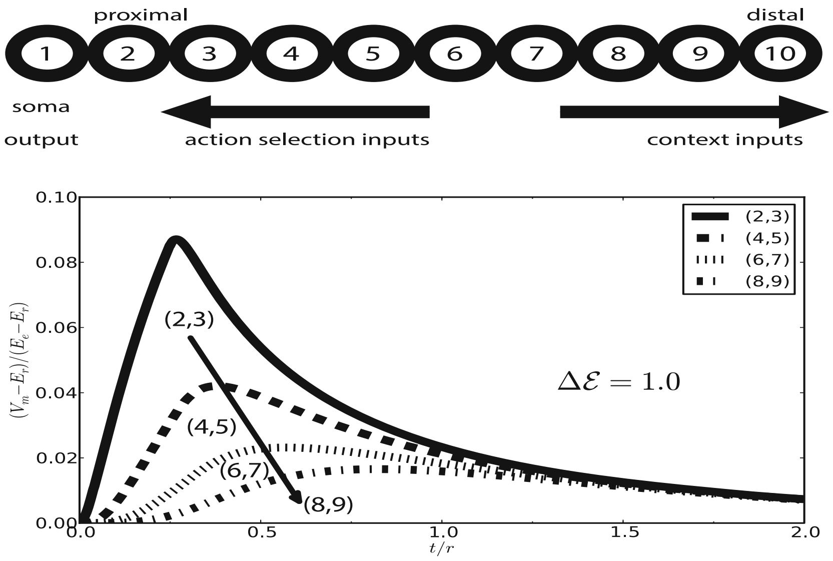


Fig. 1. Illustration of the Rall model: our reproduction of Fig. 6 from [4] shows the effect of dendritic location upon excitatory $\mathcal{E}$-pulse depolarization ($\Delta\mathcal{E} = 1.0$ and $\Delta t/\tau = 0.25$)

where C_m is the membrane capacitance, and G_e and G_i are synaptic conductances with corresponding reversal potentials E_e and E_i. The conductance G_r and reversal potential E_r model leak currents. We define two new variables $\mathcal{E}$ and $\mathcal{I}$,

$$\mathcal{E} = \frac{G_e}{G_r}, \ \mathcal{I} = \frac{G_i}{G_r}, \tag{2}$$

which are unit-less ratios of excitatory and inhibitory synaptic conductances. 2) Building upon the circuit model, Eq. 1, the compartmental model of a dendritic tree is an equivalent cylinder consisting of a system of 10 differential equations, each representing the dynamics of the membrane potential for one compartment. The membrane potential V_m^i for the i-th compartment satisfies

$$\tau_m \frac{dV_m^i}{dt} = -(1 + \mathcal{E}^i + \mathcal{I}^i + \Delta Z^2)V_m^i + \Delta Z^2(V_m^{i-1} + V_m^{i+1}) + \mathcal{E}^i, \tag{3}$$

where the compartmental electrotonic distance ΔZ is set to 0.2 and the membrane time constant is $\tau_m = \frac{C_m}{G_r}$. Boundary conditions are $V_m^{i=0} = V_m^{i=11} = 0$.

3 Simulation Results

3.1 Strong Distal Activation Evokes Up States

Recently, Plotkin et al. [3] used the technique of laser uncaging of glutamate to stimulate dendrites at different distances from the soma. They investigated under which conditions such dendritic stimulation, which mimics synaptic stimulation, gives rise to up state transitions as recorded intracellularly at the soma [7]. From their study we extracted **four important observations**: i) the somatically recorded membrane potentials triggered by both proximal and distal stimulation were similar in *amplitude*, ii) the peak somatic potential triggered by distal uncaging was *delayed* by approx. 50 ms compared to the peak triggered by proximal uncageing, iii) the somatic potential at the soma *decays* back to the resting voltage at almost the same time for both proximal and distal stimulation, and iv) the duration of up state was *longer* when triggered by distal compared to proximal stimulation (the time between the end of uncaging and a 50% voltage fall; see Fig.1c,d in [3]).

We first consider observation i): How can the somatic membrane potential rise to the same peak amplitude for both proximal and distal stimulation in a Rall model? We determined the location-dependence of the strength of the excitatory stimulation necessary to achieve this. Fig. 2a shows the somatic voltage in response to a proximal (dashed line) and distal stimulation (solid line). The strength of proximal stimulation was set to match experimental findings. In [3], an amplitude of 20 mV was shown to be achieved by stimulating on average 25 spines, but the threshold for evoking a sustained somatic depolarization was only about 11 spines from the distal region. Therefore, we selected 9 mV potential as the reference amplitude in this study, which is still consistent with the threshold observation (Plotkin [3], personal correspondence). As expected from a Rall model, the distal needs to obtain stronger stimulation than the proximal. In Fig. 2a, a distal stimulation of approx. 10$\times$ the strength of the proximal was required to reach the same peak amplitude at the soma (2.0 vs. 19.0).

While observation ii), the delayed peak of somatic potential, follows naturally from the Rall model, it is not compatible with observation iii). The potential decays later to rest for distal stimulation (Fig. 2a). Therefore, we determined the inhibitory distal stimulation necessary so that the somatic voltage decays back for distal stimulation at the same time as for the proximal one. Fig. 2b shows the result when applying a strong distal inhibitory stimulation after the somatic potential peaked. Thus, we have shown that key properties of up state can be produced within a Rall model. Our proposal is therefore an alternative to voltage-dependent channels in dendrites, but it requires a properly matched inhibition at distal sites.

3.2 Revisiting Directional Sensitivity in the Rall Model

We argue that Rall's most significant contribution in his 1964 paper [4] was the discovery that a model neuron is sensitive to the temporal sequence of inputs and can act as a device computing the direction of motion (distal to proximal vs. proximal to distal). This phenomenon was then later explored by [6] and very recently further supported experimentally by stimulation of dendrites [1]. **Two important observations** can be made from the study by Branco et al.

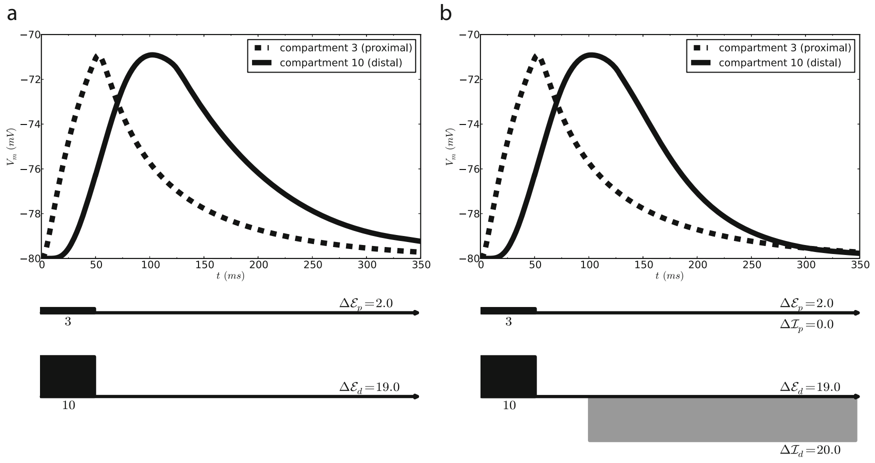


Fig. 2. Rall's model captures the key features of the up-state transition observed in striatal spiny neurons. **a)** Somatic response evoked by placing on compartment 3 an $\mathcal{E}$-pulse with magnitude $\Delta\mathcal{E}_p = 2.0$ (dashed line), and on compartment 10 an $\mathcal{E}$-pulse with magnitude $\Delta\mathcal{E}_d = 19.0$ (solid line). The latter value was chosen such that the two peak responses are the same (here: approx. -71 mV). These simulations are consistent with observations i) and ii) but not observation iii). **b)** Same as in a), but with additional delayed distal inhibition, which makes the model prediction consistent with all four observations.

[1], which used so-called IN and OUT protocols[1]: v) The IN protocol produces a larger maximal somatic peak potential than the OUT protocol (by about 35%). vi) The somatic potential evoked by the OUT protocol is higher than for the IN protocol in the decay phase.[2]

We first simulated IN and OUT sequences using Rall's original protocol (reproducing his Fig. 7 in [4]), shown in Fig. 3a. Here, the stimulation amplitude at each compartment is the same. We observe that the peak potential in the IN protocol is higher than in the OUT protocol, which constitutes the direction-selectivity of Rall's model, but the value is almost twice as high for OUT sequence and not around 35%. Furthermore, the somatic voltage in the decay phase is *not* higher in OUT protocol as observed experimentally [1].

We then simulated the Rall model, taking the strengths of the excitatory stimulation so that the somatic peak amplitudes are the same for all stimulation locations (Fig. 2a). We find that now the model predictions are in agreement with observations v) and vi). More specifically, the IN protocol produces a larger response at the soma peaking at -58.5 mV, which is 5.5 mV above the peak potential obtained by the OUT protocol (-64 mV), well matching a 35% increase. Furthermore, the OUT sequence now produces a larger potential in the

[1] The two protocols represent the sequential activation of compartments from the distal to the somatic (IN) or from the somatic to the distal (OUT) locations.

[2] We continue the numbering from the observations i-iv) in Plotkin et al. [3].

decay phase than the IN sequence. This is due to the overall stronger excitatory activation in our protocol. Note that we derived the location-dependence of the stimulation strength to evoke the same peak amplitude at the soma (Fig. 2a), and not to match the direction-selectivity reported in [1] (Fig. 3b).

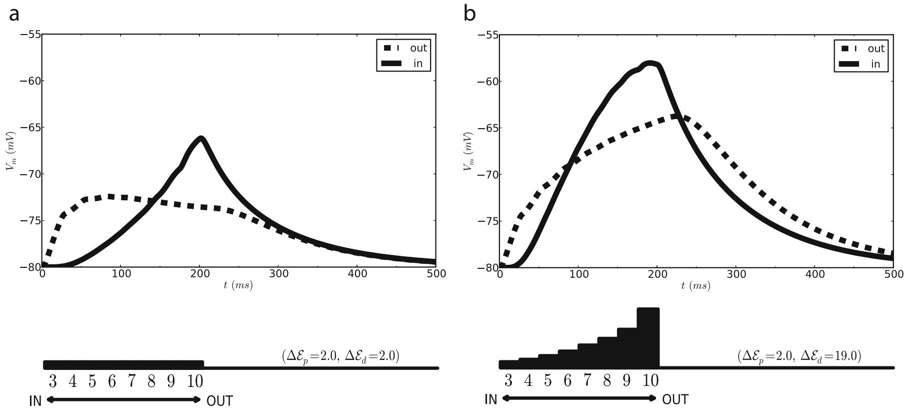


Fig. 3. Direction-selectivity in the Rall model. **a)** Stimulation protocol as used in Rall's original paper [4] (his Fig. 7). While this established dendrites as being able to compute direction-selectivity, it is *not* consistent with the recent findings from [1], observations v-vi). **b)** Simulations of IN and OUT sequences, but with stimulation strengths at each compartment that would evoke the same somatic peak response when applied alone (see Fig. 2a for the stimulation amplitudes for the two compartments 3 and 10). These simulations are consistent with recent experimental findings[1], observations v-vi).

3.3 Background Depolarization Set by Distal Inputs

It was recently suggested that context-dependent pattern recognition can be realised by synaptically driven state transitions, i. e. distal stimulation of dendrites triggers a transition to an up state, where pattern recognition of activations at proximal sites can be performed [3]. Here we explore if such a computation can also be performed in a Rall model, when the distal activation is strong as we deduced it to match properties of up state (Fig. 2a).

We simulated three scenarios (see Fig. 4): *First*, using the exact parameters as in [4], two distal $\mathcal{E}$-pulses were delivered before one proximal $\mathcal{E}$-pulse (thin solid line). The strength of all pulses was the same as in Rall's original work [4]. The obtained somatic peak potential is slightly above -70 mV (Fig. 4, thin grey line), which is far from the voltage threshold for up state transition (-60 mV, see [7]). *Second*, three consecutive proximal $\mathcal{E}$-pulses were applied, all of them with the same strength as in the first scenario (dashed line). Interestingly, the peak potential at the soma is only a few millivolts higher than in the first case. *Third*, similar to the first scenario, we simulated two distals followed by a proximal stimulation (thick solid line). Now, however, we used the stimulation strengths we determined, namely with stronger activation for distal sites. This gives rise to a much larger depolarization than in the previous two cases. The soma is depolarized by 20 mV, close to the spike threshold −60 mV.

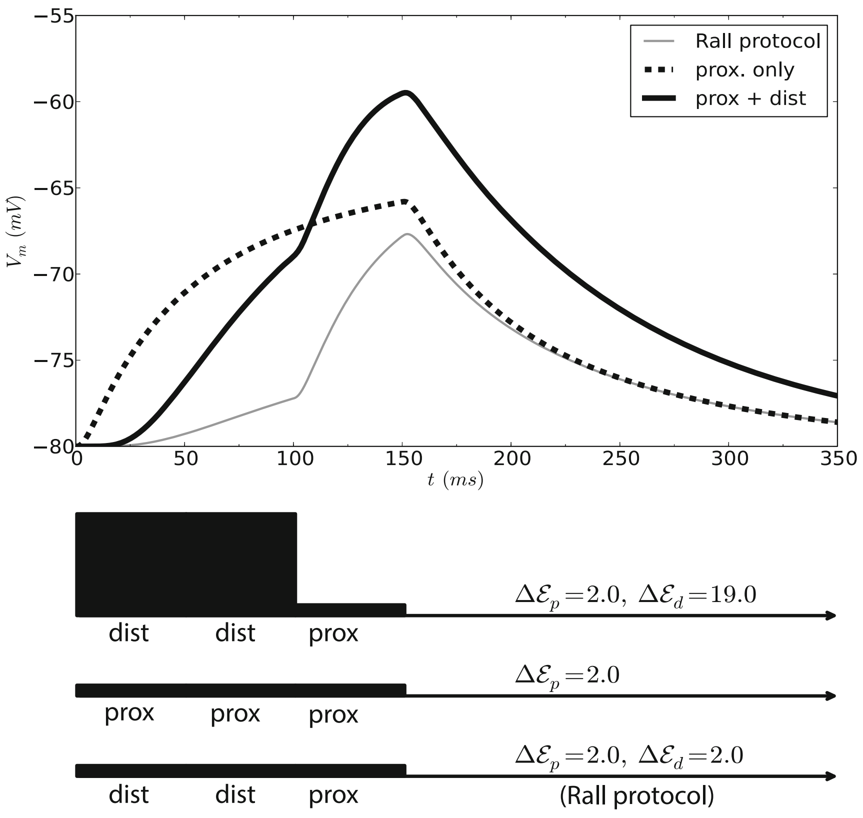


Fig. 4. Effects of sequential distal/proximal depolarization on somatic response; **prox + dist**: first strong distal, then weak proximal stimulation (solid black line); **prox only**: only weak proximal stimulation (dashed black line); **Rall protocol**: first weak distal, then weak proximal stimulation (solid grey line)

4 Discussion

In this paper we re-investigated dendritic computations in a Rall model, i. e. with passive dendrites. Extending Rall's original work we considered the location-dependence of synaptic activations as free parameters, which were determined to match recorded somatic membrane potentials in striatal neurons undergoing synaptically driven transitions to an up state.

We find that if distal is stronger than proximal activation, the Rall model predicts key features of the up state, which so far have been hypothesized to involve voltage-dependent channels in dendrites [3]. Assuming also a delayed and strong distal inhibition further improves this match as now the decay to resting voltage happens at the same time, independent of stimulation location. Besides accounting for key features of the up state, and hence providing the substrate for context-dependent pattern recognition, the strong distal activations also make the Rall model's predictions very similar to recent experimental findings on direction selectivity as computed by dendrites [1].

Our results show that dendritic computations, previously thought to require active dendrites, can be realized with passive dendrites. This shall not rule out or even replace models with voltage-dependent dendritic channels. As a consequence, future experimental studies have to determine on a case-by-case basis the actual implementation of dendritic computations and distinguish between the single-cell model with active channels, properly matched location-dependent synaptic strengths, or a mixture of both.

Acknowledgement. We thank Dr. Joshua L. Plotkin for discussion on the experimental data. This work is supported by German Research Foundation (DFG).

References

1. Branco, T., Clark, B.A., Häusser, M.: Dendritic discrimination of temporal input sequences in cortical neurons. Science 329(5999), 1671–1675 (2010)
2. Magee, J., Cook, E.: Somatic epsp amplitude is independent of synapse location in hippocampal pyramidal neurons. Nat. Neurosci. 3(9), 895–903 (2000)
3. Plotkin, J.L., Day, M., Surmeier, J.D.: Synaptically driven state transitions in distal dendrites of striatal spiny neurons. Nat. Neurosci. 14(7), 881–888 (2011)
4. Rall, W.: Theoretical significance of dendritic trees for neuronal input-ouput relations. In: Reiss, R., Alto, P. (eds.) Neural Theory and Modeling. Standford University Press (1964)
5. Rall, W.: Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic input. J. Neurophysiol. 30(5), 1138–1168 (1967)
6. Torre, V., Poggio, T.: A synaptic mechanism possibly underlying directional selectivity to motion. Proc. R. Soc. Lond. 202(1148), 409–416 (1978)
7. 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)

Modeling Action Verb Semantics Using Motion Tracking

Timo Honkela[1] and Klaus Förger[2]

[1] Department of Information and Computer Science
[2] Department of Media Technology
Aalto University School of Science, FI-00076 Aalto, Finland

Abstract. In this article, we consider how semantics of action verbs can be grounded on motion tracking data. We present the basic principles and requirements for grounding of verbs through case studies related to human movement. The data includes high-dimensional movement patterns and linguistic expressions that people have used to name these movements. We discuss open issues and possibilities related to symbol grounding. As a conclusion, we find the grounding to be useful when reasoning about the meaning of words and relationships between them within one language and potentially also between languages.

1 Introduction

The basic scientific question behind this article is how to computationally model the interrelated processes of interpreting natural language and perceiving movement in multimodal real world contexts. Namely, an important problem in natural language processing and in computer science in general is that in most cases computer systems processing symbols or language do not have access to the phenomena being referred to. In contrast, humans can readily associate expressions with their non-linguistic experiences and actions in the world. For instance, we know the different interpretations of color red in expressions “red skirt”, “red skin” and “red wine” or the phrase “a long jump” may refer to very different things depending on the context. As a direct consequence, computational systems can only reason about the symbols themselves rather than about the grounded meaning or external references of those symbols. However, if we want machines to learn and use language as it is actually used by humans, we have to take into account that language is fully understood only through its use in linguistic and multimodal contexts [1].

In this article, we consider a seemingly simple domain of symbol grounding, naming human movement. It is, however, complex enough to be a non-trivial case which is also illustrated by the fact that different languages divide the space of body-related expressions in different ways [2]. Moreover, people may have different interpretations even regarding what they call “running”, “jogging” or “walking” in less prototypical cases. Studying these differences is enabled by having access to the actual patterns of movement.

Extracting semantic information from the statistical regularities in large text corpora is nowadays commonplace. One obvious reason for using a statistical approach is

© Springer-Verlag Berlin Heidelberg 2013

cost-effectiveness: models of language can be built with less human effort than when traditional means are used. While statistical analysis of word occurrences and other linguistic constructions in their textual contexts has proven to be useful, there is a limit to how much can be inferred from texts only, and therefore obtaining data of words in their multimodal contexts is an important research topic. This kind of external contextualization is often referred to as symbol grounding [3].

Research on symbol grounding is multidisciplinary and multifaceted. Related to motion tracking, successful systems that achieve good classification results include identification of movement types [4], manners of moving [5] and the gender from walking movements [6]. The issue is, of course, relevant in robotics [7,8]. In cognitive science, symbol grounding and embodiment is an important theme (cf., e.g., [9,10,11,12,13]). In a classical work, Bailey developed a computational model of the role of motor control in the acquisition of action verbs [14].

We are aware of the breadth and depth of the underlying philosophical [15] and methodological issues. In this article, we wish to address naming human movements as a concrete, limited but non-trivial case related to multimodally grounded natural language processing. In order to study how people name different human movements, we have used motion tracking to obtain data in which skeletons move on the screen. Using this data, we conducted two case studies. In the first study, we asked people to classify movements to a limited number of categories. The results of this classification task, serving as a feasibility study, are reported in the next section.

In the second case study, we asked people to describe these movements with their own words. It was important that the question was open ended because we wished to study the naming of movements which is different from classification. In naming, the labels given typically follow a Zipfian distribution [16]. The results of this case study are reported discussed in Section 3.

2 Grounding through Motion Capture

The motion tracking has been conducted using OptiTrack Motion Capture system and the ARENA software, developed by NaturalPoint, Inc. We recorded 16 minutes of human motion which was manually annotated with the following labels: jump, sit down, sitting, stand up, standing, turn left, turn right, walking and waving hand. The labels were allowed to overlap as for example walking and waving hand can be done at the same time.

Four types of features were extracted from the data (see Figure 1). The first type was absolute values of velocities of all the body parts. The second type was distances between the end parts of the limbs. The third was velocity vectors of the end parts of the limbs. The last type was coordinate positions of the end parts of the limbs. To make the velocity vectors and positions usable, we had to center the coordinate system to the hips of the character and rotate the character to always face the same direction. This resulted in 72 feature dimensions in the first case study. We averaged the values of the features over 0.25 seconds to get the final values used in the classification.

Grounding verbs requires associating them with patterns of motion based features. A good way to ensure that the used features are not only random numbers is to see how well the features can be used in classifying previously unseen motions. To classify the

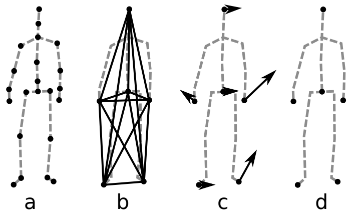


Fig. 1. The types of the features used in the classification include absolute velocities for each body part (a), distances between limb ends (b), velocity vectors of limb ends (c), and positions of limb ends (d)

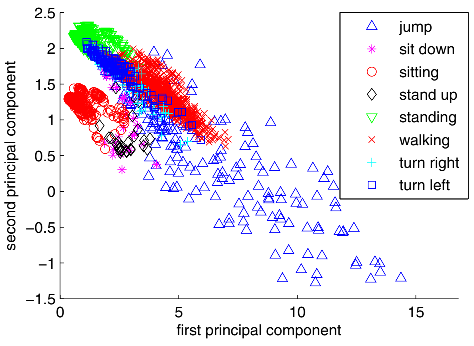


Fig. 2. Training samples used the classification plotted along the first two principal components

data we used K nearest neighbors with a Euclidean distance metric. The classification was tested on two minutes of motion that was not used in the training set with results at the same level as obtained earlier by others [4,5,6]. In classification, the transition motions between two verbs were the main problem. The classifier tried to forcible classify the motion when the most natural option would be not giving a class at all as the transitions may not correspond to any verb. One reason for the good performance lies in the well-selected features. When the training samples are plotted along the first two principal components (see Figure 2), it becomes evident that many of the classes are separated by the used features.

The features form a space where all individual frames of a motion can be projected. As two consecutive frames of motion are always similar due to physical restrictions, motions can be plotted as trajectories in the feature space. This is shown in Figure 3, where a motion starting from sitting, going through standing up, standing and walking, is plotted along the first two principal components of the feature space.

The separation by the two principal components is not complete as the data is inherently high dimensional as can be seen in Figure 4. The figure shows that more than 10 principal components are needed in order to explain 90% of the variance in the data.

The fact that many labels can be valid for a motion simultaneously is a challenge for using the features of the classification as distance measures. For example, waving hand can be done while walking, sitting or standing. This is visible in the training samples used for those classes in Figure 5. As 'waving hand' appears in several separate clusters, the mean distance between it and other labels does not reflect the real relations between the labels. Therefore, the overlap between labels should be analyzed before similarity of the labels.

3 Modeling Relations between Verbs

In order to have a fine-grained collection of movements, we asked actors to perform walking, limping and running in various styles. These movements were blended in three steps including alignment, time warping and interpolation. In time warping, the motions

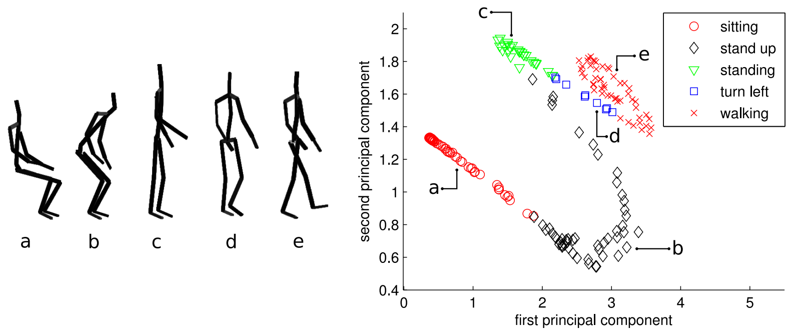
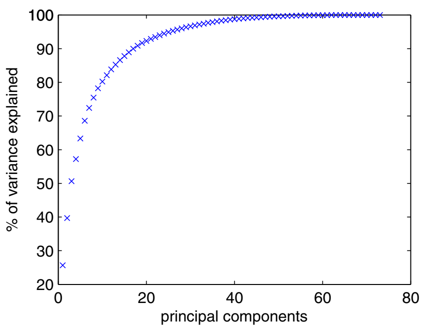


Fig. 3. On the left hand side, motion of a character (a) sitting, (b) standing up, (c) standing, (d) turning left and (e) walking (left), and on the right hand side, the trajectory formed by the frames plotted on the first and second principal component



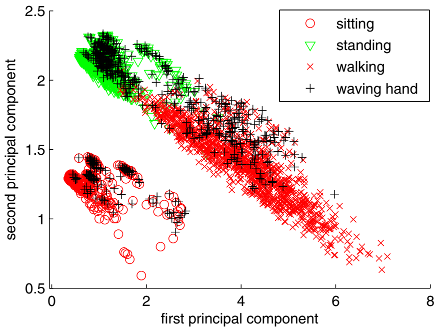


Fig. 4. Variance explained by the principal components plotted cumulatively

Fig. 5. Training samples with label 'waving hand' and other samples with labels that can be used simultaneously plotted along the first two principal components

were synchronized. In the third step, the coordinates of the root joints were interpolated linearly and the joint rotations were interpolated in a four-coordinate system [17]. The blending enabled us to have a larger number of variations of the movements. The people were asked to describe the movement with one verb or phrase. The task was to label 24 to 124 videos where the videos lasted from 3 to 12 seconds. Each video was portraying a stick figure representation.

We analyzed the questionnaire results where 22 persons had named the movements in Finnish language. We used the self-organizing map (SOM) [18] algorithm to conduct a mapping from the 602-dimensional movement space into a 2-dimensional display. The movement determines the map structure and structure of labels is obtained by including them in the input vector with a small weight. To illustrate the outcome, we chose 12 verbs to be analyzed in more detail. This map of labels is shown Fig 6. With one exception, each verb is associated with a contiguous area on the map. For instance, the verb

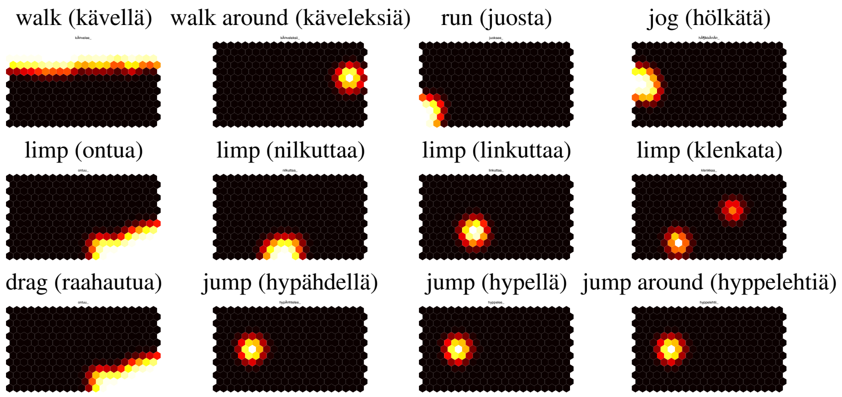


Fig. 6. The distribution of 12 verbs on a self-organizing map of movements. A light shade denotes a high value of each verb.

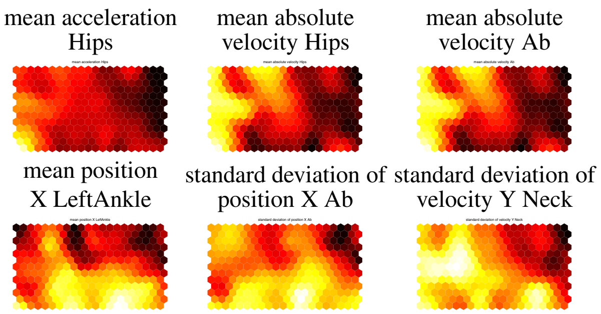


Fig. 7. The distribution of 6 features out of 602 on the self-organizing map of movements. A light shade denotes a high value of each feature.

"walk" is located on the upper side of the map and the verb "run" on the lower left corner. Fig 7 shows examples of underlying movement features that have determined the organization of the map. The area for running in Fig 6 coincides with the feature "mean acceleration of hips" in Fig 7. The union "running" and "jogging" coincides with the distribution of features "mean absolute velocity of hips" and "mean absolute velocity of abdomen".

In many cases, the association between the labels and patterns of movement is not one-to-one but require consideration of a reasonably large number of features. On the other hand, the vector space for the associations is much lower in dimensionality than the pixel patterns over time in the original videos. This is thanks to the motion tracking system that compresses the original very high dimensional feature space into a large number of meaningful variables. In the general case, it remains a challenge how to conduct the pattern recognition and dimensionality reduction in such a way that relevant features are included for the associations. In many early studies the low-level representations were based on manually encoded structures (cf., e.g., [14]). In order to develop large scale solutions, the process should, however, be as automatic as possible. Due to variety of applications that may require different kinds of feature sets for same domain, the features extraction process needs to be task-dependent [19].

4 Conclusions and Discussion

We are interested in answering the scientific question of how to enable machines to have a increasingly common ground with humans for associating language with perceptual patterns. In the following, we discuss two symbol grounding themes.

4.1 Multimodally Grounded Language Technology

Through multimodally grounded language technology, more robust and correct manipulation of linguistic data becomes possible, e.g., when resolving ambiguities or when needing deeper inference. Application areas include building animations using linguistic instructions and coaching of skills.

What centrally constrains communication is the dissimilarity of the conceptual systems of the discussants. An important aspect of better understanding of human expression is, we believe, capturing human conceptualizations of the environment in which the co-operation is to take place. The subjective aspect of interpretation can be analyzed when the use of symbols is considered in different contexts by a number of individuals [20].

4.2 Multimodally Grounded Translation

It has earlier been demonstrated that the association with visual information can be used even to find parallels between different languages [21]. An analysis of the similarities in the visual appearance of some object can be used to find a conceptual link between a word in one and another language. This is analogical to ostensive definition, based on pointing out examples. In the future, we plan to collect labeled data in multiple languages. This enables developing a mapping function between action verbs in different languages based on the common ground.

Acknowledgments. We gratefully acknowledge the financial support by Academy of Finland for the Multimodally Grounded Language Technology project (254104) and HeCSE graduate school that have made this research possible. We also wish to thank our colleagues in the project who have helped in conducting the research in various ways including Prof. Tapio Takala, Dr. Jorma Laaksonen, Dr. Markus Koskela, Dr. Harri Valpola, Dr. K. Lagus, Ms. Xi Chen, Mr. Paul Wagner and Mr. Oskar Kohonen. The insights in this area have also progressed thanks to collaborations with Prof. Peter Gärdenfors (Lund University, Sweden) and Prof. Lars Kai Hansen (DTU, Denmark) and their teams.

References

1. Hörmann, H.: Meaning and Context. Plenum Press, New York (1986)
2. Choi, S., Bowerman, M.: Learning to express motion events in English and Korean: The influence of language-specific lexicalization patterns. Cognition 41(1), 83–121 (1991)
3. Harnad, S.: The symbol grounding problem. Physica D 42, 335–346 (1990)
4. Pavlović, V., Rehg, J., MacCormick, J.: Learning switching linear models of human motion. In: Advances in Neural Information Processing Systems 13, pp. 981–987 (2001)
5. Taylor, G.W., Hinton, G.E., Roweis, S.: Modeling human motion using binary latent variables. In: Advances in Neural Information Processing Systems 19, pp. 1345–1352 (2007)
6. Davis, J., Gao, H.: Gender recognition from walking movements using adaptive three-mode PCA. In: Computer Vision and Pattern Recognition Workshop (2004)
7. Roy, D.: Grounding words in perception and action: computational insights. Trends in Cognitive Sciences 9(8), 389–396 (2005)
8. Williams, M.A., McCarthy, J., Gärdenfors, P., Stanton, C., Karol, A.: A grounding framework. Autonomous Agents and Multi-Agent Systems 19(3), 272–296 (2009)
9. Lakoff, G., Johnson, M.: Philosophy in the Flesh - The Embodied Mind and its Challenge to Western Thought. John Wiley, New York (1999)
10. Glenberg, A.M., Robertson, D.A.: Symbol grounding and meaning: A comparison of high-dimensional and embodied theories of meaning. Journal of Memory and Language 43(3), 379–401 (2000)
11. Sun, R.: Symbol grounding: a new look at an old idea. Philosophical Psychology 13(2), 149–172 (2000)
12. Vogt, P.: The physical symbol grounding problem. Cognitive Systems Research 3(3), 429–457 (2002)
13. Gärdenfors, P., Warglien, M.: Using conceptual spaces to model actions and events. Journal of Semantics 29(4), 487–519 (2012)
14. Bailey, D.: When Push Comes to Shove: A Computational Model of the Role of Motor Control in the Acquisition of Action Verbs. PhD thesis, UC Berkeley (1997)
15. Honkela, T.: Philosophical aspects of neural, probabilistic and fuzzy modeling of language use and translation. In: Proceedings of IJCNN 2007, pp. 2881–2886 (2007)
16. Li, W.: Zipf's law everywhere. Glottometrics 5, 14–21 (2002)
17. Shoemake, K.: Animating rotation with quaternion curves. SIGGRAPH Computer Graphics 19(3), 245–254 (1985)
18. Kohonen, T.: Self-Organizing Maps. Springer (2001)
19. Ji, R., Yao, H., Liu, W., Sun, X., Tian, Q.: Task-dependent visual-codebook compression. IEEE Transactions on Image Processing 21(4), 2282–2293 (2012)
20. Honkela, T., Raitio, J., Nieminen, I., Lagus, K., Honkela, N., Pantzar, M.: Using GICA method to quantify epistemological subjectivity. In: Proc. of IJCNN 2012, pp. 2875–2883 (2012)
21. Sjöberg, M., Viitaniemi, V., Laaksonen, J., Honkela, T.: Analysis of semantic information available in an image collection augmented with auxiliary data. In: Proc. of AIAI 2006, Artificial Intelligence Applications and Innovations, pp. 600–608. Springer (2006)

Evolution of Dendritic Morphologies Using Deterministic and Nondeterministic Genotype to Phenotype Mapping

Parimala Alva[1], Giseli de Sousa[1,2], Ben Torben-Nielsen[3], Reinoud Maex[4], Rod Adams[1], Neil Davey[1], and Volker Steuber[1]

[1] STRI, University of Hertfordshire, Hatfield, UK
{p.alva2,r.g.adams,n.davey,v.steuber}@herts.ac.uk
[2] Federal University of Santa Catarina, Brazil
[3] École Polytechnique Fédérale de Lausanne, Switzerland
[4] École Normale Supérieure, France

Abstract. In this study, two morphological representations in the genotype, a deterministic and a nondeterministic representation, are compared when evolving a neuronal morphology for a pattern recognition task. The deterministic approach represents the dendritic morphology explicitly as a set of partitions in the genotype which can give rise to a single phenotype. The nondeterministic method used in this study encodes only the branching probability in the genotype which can produce multiple phenotypes. The main result is that the nondeterministic method instigates the selection of more symmetric dendritic morphologies which was not observed in the deterministic method.

Keywords: pattern recognition, evolutionary algorithm.

1 Introduction

A variety of neurons are present in the brain and different types of neurons exhibit distinct dendritic morphologies. Recent work has shown that dendritic morphologies affect the back propagation of action potentials, synaptic integration and other aspects which have implications for the proper functioning of the neuron[5,7]. However, a clear understanding of the computational implications of dendritic structure does not exist. One approach to try to understand how the dendritic structure is related to function is to optimize the dendritic morphology of a neuronal model for a particular task using an Evolutionary Algorithm. The best individuals produced as a result of the evolution can act as an indicator as to which features of the dendritic morphology are important for that particular function.

In previous work, de Sousa[2] presented an algorithm, hereafter referred to as the de Sousa algorithm, where the dendritic morphology best suited for a pattern recognition task was studied. The purpose of the present study is to verify if the results obtained by the de Sousa algorithm are not merely a consequence

© Springer-Verlag Berlin Heidelberg 2013

of the way the algorithm is implemented and if they do indeed show general properties of neurons which are best suited for pattern recognition. This is done by applying a different algorithm, the Torben-Nielsen algorithm[1] which uses a nondeterministic method of genotype to phenotype mapping as opposed to the deterministic genotype to phenotype mapping used in the de Sousa algorithm, to the same task of pattern recognition in neurons. It is interesting to see if the two different methods for specifying morphology in the genotype highlight the same features of a dendritic morphology as being important for pattern recognition.

2 Pattern Recognition in Neurons

The pattern recognition capability of neurons enables them to distinguish between learned and novel patterns. In this study, a simple one-shot Hebbian Learning[9] has been used in the neuronal models: if the N binary patterns to be learned are $\boldsymbol{x}^{\mu}$ ($\mu\ is\ 1..N$), then the weight at synapse i is given by $w_i = \sum_{\mu} x_i^{\mu}$. In the recall phase, a set of novel patterns are presented along with the stored patterns. As a consequence of the change in the synaptic weights, the neuronal model produces a higher amplitude of Excitatory Post Synaptic Potentials (EPSPs) for the stored patterns than for the novel patterns (see [8]).

3 Methods

The steps followed by the two algorithms, the de Sousa algorithm and the modified Torben-Nielsen algorithm, are depicted in Fig. 1. We use the term 'modified' because the original algorithm developed by Ben Torben-Nielsen to evolve neuronal models for co-incidence detection could not be used as-is for comparison with the results of the de Sousa algorithm. Some modifications were necessary to ensure a fair comparison between the two algorithms. The modifications made to the algorithm are listed here:

1. The algorithm was modified to generate fixed size trees. The number of terminal nodes, length, diameter, tapering of the compartments, and other parameters relating to the pattern recognition task were set to the values used in the de Sousa algorithm.
2. In the original algorithm, the branch probability at every bifurcation point was a function of the distance of the bifurcation point from the soma of the neuronal model. This was changed and a new method (see Section 3.1) was implemented to give us more variation in the genotypes produced.
3. The ability of the branch to bifurcate, extend or terminate at every branch point was restricted to bifurcate or terminate in order to produce trees with fixed sized compartments.
4. The original algorithm ensures that a single genotype maps to a single phenotype by fixing the random seed used in each optimization run. In the modified version of the algorithm, a single genotype can map to a range of phenotypes.

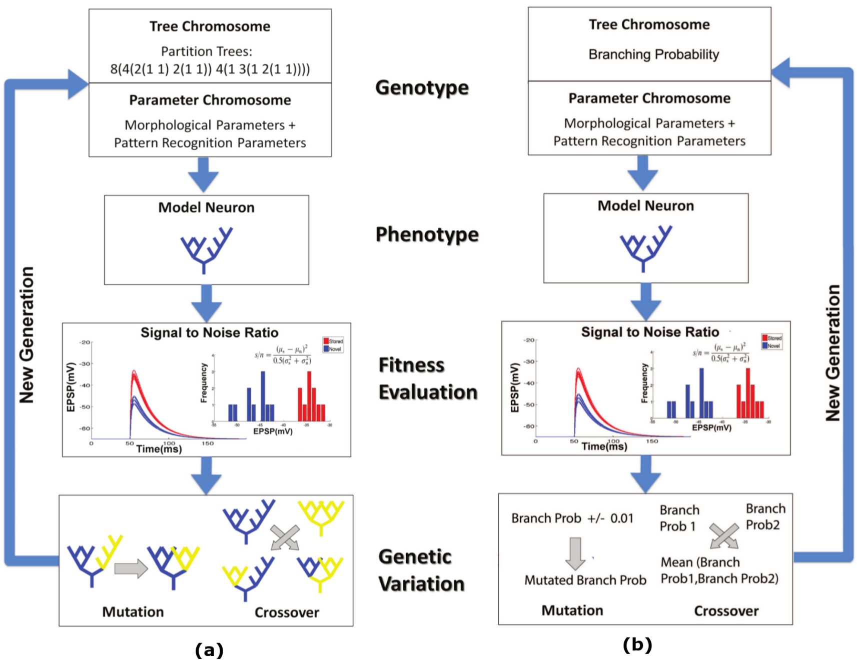


Fig. 1. Steps of the evolutionary algorithm followed by the de Sousa Algorithm (a) and the modified Torben-Nielsen algorithm (b). The de Sousa Algorithm represents the morphology of the neuronal model as partitions in the genotype while the modified Torben-Nielsen algorithm encodes only the branching probability in the genotype.

A detailed description of each step of the Evolutionary Algorithm pertaining to the de Sousa algorithm and the modified Torben-Nielsen algorithm is presented in the following sub-section.

3.1 Genotype Representation

The de Sousa algorithm encodes the exact branching pattern of the dendritic tree in the genotype. The branching pattern of the phenotype is represented as a set of partitions using the method proposed by Van Pelt and Verwer [3].

The modified Torben-Nielsen algorithm uses a nondeterministic method of genotype to phenotype mapping. In this method, the morphology of the entire dendritic tree is specified only by a branching probability, a number between 0 and 1, in the genotype. Other morphological parameters similar to the de Sousa algorithm may also be encoded in the genotype but in the results presented here we kept them fixed.

Algorithm 1. The modified Torben-Nielsen Algorithm for genotype to phenotype mapping.

```
while(number of terminal nodes < 128)
{
    traverse the bottom ply of the tree from left to right
    for(each node visited)
    {
        bifurcate or terminate according to branching probability
    }
    if(no node chooses to bifurcate)
    {
        force the last node visited to bifurcate
    }
 }
```

3.2 Genotype to Phenotype Mapping

The genotype to phenotype mapping in the de Sousa algorithm is fairly straightforward. The partitions in the genotype dictate the morphology of the phenotype. At every bifurcation point, a partition specifies the number of terminal nodes in the right and left sub-tree. In the modified version of the Torben-Nielsen algorithm, the branching probability which is part of the genotype affects the morphology of the phenotype produced. For example, a branching probability value of 0.75 means that at every branch point, the branch has 0.75 probability of bifurcating and 0.25 probability of terminating. Different branching probabilities give rise to different types of dendritic trees in terms of asymmetry and mean depth. The asymmetry index as given by Van Pelt et al.[4] indicates the overall shape while the mean depth metric [2] indicates the average distance between the synapse and the soma of the neuronal model. Figure 2b shows the variation of asymmetry index and mean depth of the phenotypes for sample branching probability values of 0.25, 0.50, and 0.75. From this figure, we can observe that the same branching probability, or the same genotype, can produce a range of phenotypes that vary in terms of asymmetry index and mean depth. Algorithm 1 shows the detailed steps followed in the generation of the phenotype from the genotype using the modified Torben-Nielsen algorithm.

Note that in Algorithm 1 the tree is likely to be skewed to the right, a desirable feature because we wanted also to have asymmetrical trees in our population. The higher frequency of dendritic trees having an asymmetry index of 0.99, as shown in Fig. 2a, is also a result of the forced bifurcation of the last visited node as shown in Algorithm 1. Figure 2b shows the variation of asymmetry index and mean depth of the phenotypes for sample branching probability values of 0.25, 0.50, and 0.75. The tree morphology which is a result of genotype to phenotype mapping, in case of both algorithms, is converted into a neuronal model using the NEURON simulator software[6]. The neuronal model has a membrane capacitance (C_m) of 0.75 μF/cm^2, an axial resistance(R_a) of 150 Ωcm,

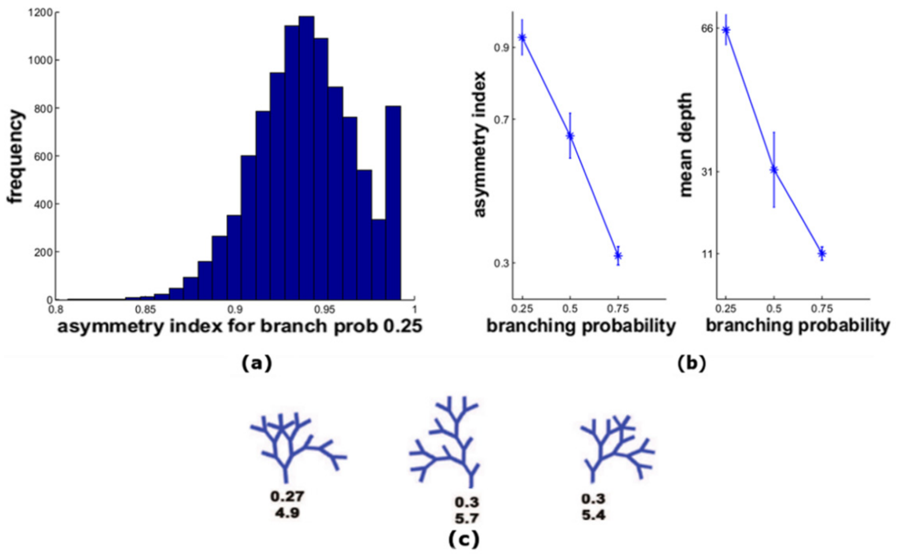


Fig. 2. (a)Histogram showing the distribution of asymmetry indices of the dendritic trees for a branching probability of 0.25 when the modified Torben-Nielsen algorithm is used. (b) Mean and Standard Deviation Graphs showing the range of the asymmetry index and mean depth for sample branching probabilities of 0.25, 0.50, 0.75.(c) Three different tree structures, each having a different asymmetry index (the first row below the tree) and mean depth (second row), produced from the same genotype having a branching probability of 0.75.

a specific membrane resistance (R_m) of 30 kΩcm^2 and a leak reversal potential of $Eleak$=-70 mV. Each compartment of the neuronal model is of equal length and diameter with no tapering. Since this study is limited to studying passive neuronal models, active conductances are not applied to the model.

3.3 Fitness Function

Since the neuronal models used in this study are passive, the response of the neuronal model to the pattern presented is measured by the amplitude of the EPSPs produced in the soma. The ability of the neuron to distinguish between stored and new patterns is determined by the signal to noise ratio of the amplitudes of the EPSP produced by stored and novel patterns (see [8]).

3.4 Genetic Variation

In case of both algorithms, a process of selection is applied to the population to select the best 10 % of the individuals to be propagated to the next generation. As is normally the case, selection, crossover and mutation are applied to the population to produce the next generation.

4 Results

To evaluate the morphology of the dendritic tree best suited for pattern recognition, we observe the change of asymmetry index and mean depth of the dendritic trees. Details of calculation of the metrics, asymmetry index and mean depth, have been given in [2].

4.1 Results of the de Sousa Algorithm

The change of values of the fitness as well as the two morphological metrics are observed over 60 generations of the de Sousa algorithm as shown in Fig. 3a. The initial population for this experiment is biased to consist mainly of asymmmetric morphologies. However, a huge drop in the asymmetry index and mean depth values are observed within the first few generations indicating that neuronal models with more symmetrical morphologies and lower mean depth perform better. However, this does not mean that the asymmetry index and fitness are inversely correlated over the whole range of asymmetry indices. After the dendritic trees achieve an asymmetry index of about 0.4, a further reduction in asymmetry does not increase the fitness of the dendritic trees. When dendritic trees with similar asymmetry index were binned together and their mean depth values evaluated as shown in Fig. 3c, it was noticed that the dendritic trees with asymmetry indices of about 0.4 and lower had very similar values of mean depth. This may explain the plateau in the fitness value after an asymmetry index of 0.4 was reached, and it suggests that perhaps mean depth is a better indicator of pattern recognition performance than asymmetry index.

4.2 Results of the Modified Torben-Nielsen Algorithm

The same experiment, where the initial population was biased to consist mainly of asymmetric morphologies, was executed for the modified Torben-Nielsen algorithm. As shown in Fig. 3b, the results of the experiment follow the same pattern as for the de Sousa algorithm in the case of fitness and mean depth values. The main difference was the change of the asymmetry index of the dendritic trees. In the de Sousa algorithm, the asymmetry index remains at a value of about 0.4 and does not reduce further as this value of asymmetry index is sufficient to produce the best individual for pattern recognition. However, with the modified Torben-Nielsen algorithm, the asymmetry index of the dendritic trees reduces to a value of around 0.2 at the end of 100 generations. This is a consequence of the nondeterministic approach used in this study, which generates a new tree instantiation each time a new individual is passed to the next generation. As shown in Fig. 2a and Fig. 2b, a single genotype can map to different phenotypes that vary in asymmetry index and mean depth. Since there is a considerable variation in the type of phenotype produced by a single genotype, there is an additional pressure on the Evolutionary Algorithm to select a genotype such that all resulting phenotypes perform well. This forces the algorithm to select genotypes which produce, on average, more symmetrical trees

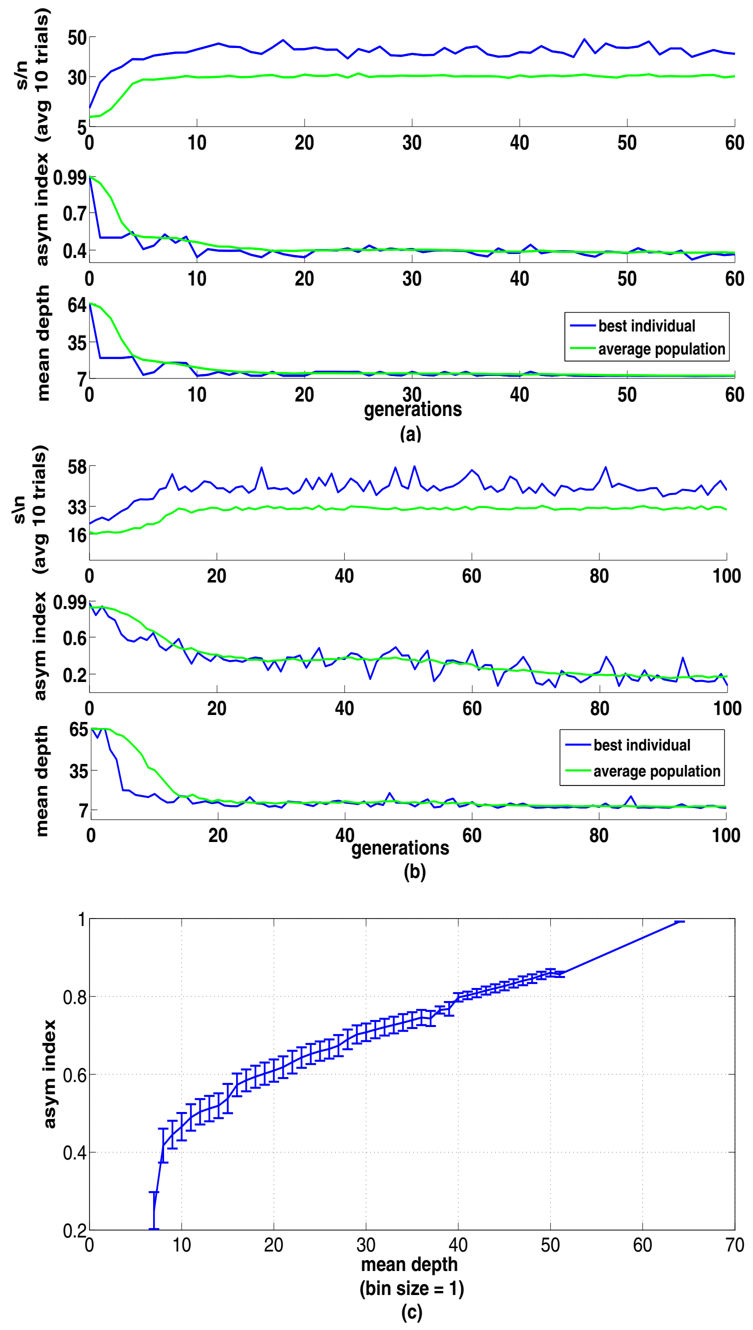


Fig. 3. (a) Change of fitness, asymmetry index and mean depth in the de Sousa algorithm. (b) Change of fitness, asymmetry index and mean depth in the Torben-Nielsen algorithm. (c) Mean depth vs asymmetry index for dendritic trees in the de Sousa algorithm.

than their counterparts which show mixed asymmetry values. The final outcome of this experiment reinforces the results produced by the de Sousa algorithm that symmetrical neuronal morphologies which have low mean depth values are well suited for pattern recognition.

5 Conclusion

Previous work [2] has shown that a Genetic Algorithm will reduce the mean depth of the dendritic tree in order to improve the pattern recognition performance. Here we demonstrate that the same principle applies with a different morphological specification, namely a nondeterministic mapping of the genotype to the dendritic tree morphology. The nondeterministic method of genotype to phenotype mapping showed that when a single genotype can map to an array of phenotypes, the criterion for the selection of genotypes was more stringent, which led to the selection of genotypes that gave rise to more symmetrical morphologies. Symmetric morphologies have a smaller mean depth and therefore lead to less variance in responses and a higher signal to noise ratio. However, in real neurons we do not always find this symmetric morphology because the morphology of a real neuron is governed by other factors such as the need to form synaptic contacts in certain locations.

References

1. Torben-Nielsen, B., Stiefel, K.M.: An inverse approach for elucidating dendritic function. Front. Comput. Neurosci. 4, 128 (2010)
2. de Sousa, G., Maex, R., Adams, R., Davey, N., Steuber, V.: Evolving dendritic morphology and parameters in biologically realistic model neurons for pattern recognition. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 355–362. Springer, Heidelberg (2012)
3. van Pelt, J., Verwer, R.: Growth models (including terminal and segmental branching) for topological binary trees. Bull. Math. Biol. 47, 323–326 (1985)
4. van Pelt, J., Uylings, H.B., Verwer, R.W., Pentney, R.J., Woldenberg, M.J.: Tree asymmetry–A sensitive and practical measure for binary topological trees. Bull. Math. Biol. 54, 759–784 (1992)
5. Spruston, N.: Pyramidal neurons: dendritic structure and synaptic integration. Nat. Rev. Neurosci. 9, 206–221 (2008)
6. Carnevale, N.T., Hines, M.L.: The NEURON Book. Cambridge University Press, Cambridge (2006)
7. Vetter, P., Roth, A., Häusser, M.: Propagation of Action Potentials in Dendrites Depends on Dendritic Morphology. J. Neurophysiol. 85, 926–937 (2001)
8. Steuber, V., De Schutter, E.: Long-term depression and recognition of parallel fibre patterns in a multi-compartmental model of a cerebellar Purkinje cell. Neurocomputing 38-40, 383–388 (2001)
9. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison-Wesley, Reading (1991)

Sparseness Controls the Receptive Field Characteristics of V4 Neurons: Generation of Curvature Selectivity in V4

Yasuhiro Hatori[1,2], Tatsuroh Mashita[1], and Ko Sakai[1]

[1] Graduate School of Information Engineering, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
{hatori,mashita}@cvs.cs.tsukuba.ac.jp, sakai@cs.tsukuba.ac.jp
[2] Research Fellow of the Japan Society for the Promotion of Science

Abstract. Physiological studies have reported that the intermediate-level visual area represents primitive shape by the selectivity to curvature and its direction. However, it has not been revealed that what coding scheme underlies the construction of the selectivity with complex characteristics. We propose that sparse representation is crucial for the construction so that a sole control of sparseness is capable of generating physiological characteristics. To test the proposal, we applied component analysis with sparseness constraint to activities of model neurons, and investigated whether the computed bases reproduce the characteristics of the selectivity. To evaluate the learned bases quantitatively, we computed the tuning properties of single bases and the population, as similar to the physiological reports. The basis functions reproduced the physiological characteristics when sparseness was medium (0.6-0.8). These results indicate that sparse representation is crucial for the curvature selectivity, and that a sole control of sparseness is capable of constructing the representation.

Keywords: shape representation, curvature, sparse coding, computational model.

1 Introduction

The cortical representation of shape is crucial in shape perception and object recognition. Physiological studies have reported that the representation of shape is constructed in the ventral stream [1-4]. Carlson et al. have reported that V4 neurons are selective to curvature and its direction, and their population activity is biased toward acute curvature. However, it has not been clarified that what coding scheme explains the characteristics of V4 neurons. Olshausen and Field have showed that learning sparse code of the natural images produces receptive field properties of V1 neurons that are (1) localized, (2) oriented, and (3) band-passed, as nicely approximated by Gabor function [5]. They applied component analysis with sparseness constraint to the natural images. Component analysis assumed that inputs can be described by linear superposition of basis functions with appropriate coefficients. The basis functions and their coefficients were learned adaptively to realize the sparse code of natural images. If V1 neurons represent natural images by sparse code, basis functions were expected to show the receptive field properties of V1 neurons. Properties of the

© Springer-Verlag Berlin Heidelberg 2013

learned bases agreed with the characteristics of V1 neurons, suggesting that sparseness is crucial for the characteristics of V1 neurons [5].

In the present study, we propose that the representation of V4 is sparse, sharing the principle same as V1, because V4 receives ascending input from V1 and V2 [6]. The fact enables us to propose that sparse representation is crucial for curvature selectivity in V4. To test the proposal, we applied the component analysis with constraint of sparseness to activities of model neurons driven by the stimuli from natural images, and investigated whether sparse representation is capable of reproducing the characteristics of V4 neurons. In order to study the dependence of computed bases on sparseness, we changed systematically the sparseness and obtained the corresponding bases. The computed bases appeared unstructured when sparseness was small (< 0.6). In contrast, they showed localized structures that appear to represent the curvature, when sparseness was larger (> 0.6). To evaluate whether these bases in fact reproduce the receptive field properties of V4 neurons, we computed selectivity of each basis in curvature and direction domain, and their population preference, in the same manner as the physiological studies [3]. Each basis function showed tuning for curvature and its direction, and population selectivity showed the bias toward acute curvature, when sparseness was medium to large (0.6-0.8). These results indicate that sparse representation is crucial for the curvature selectivity in V4.

2 Methods

2.1 Stimuli

Stimuli were comprised by 80,000 patches whose size was 33x33 pixels. These patches were part of natural images that were taken from Amsterdam Library of Object Images [7]. The patches were made by following steps. All images were binarized to focus solely on information of the shape. After that, we cut out small patches along the contours of objects to assure that contour passed through the center of given patch.

2.2 The Model

Processing flow of the model was illustrated in Fig. 1. Component analysis with sparseness constraint was applied to responses of model V2 neurons driven by the natural images because V4 neurons receive ascending input from V2 [6]. Receptive field properties of model V2 neurons were realized by the combination of two Gabor filters, mimicking angle selectivity reported by physiological study [2].

The algorithm of the component analysis with sparseness constraint was the same as Olshausen and Field [5]. In brief, it assumed that the inputs were represented by the linear superposition of basis functions with appropriate coefficients. The goal of sparse coding was to seek adaptively a set of basis functions that effectively describe the original images with the strong activation of a small number of active bases. Therefore, achieving the sparse code is equivalent to require the cost function comprised of (1) high sparseness of coefficient, as well as (2) low error in reconstruction. The cost function was formulized as below [5]:

$$\text{cost} = [Reconstruction\ error] + \lambda[denseness],$$

$$= \sum_{x,y,j}\left[X_j(x,y) - \sum_i s_{ij}a_i(x,y)\right]^2 + \lambda\sum_{i,j} S\left(s_{ij}/\sigma\right) \tag{1}$$

, where, (x,y) represents the pixel of the image, X_j represents activity of model V2 neurons to jth image, a_i represents the ith basis function, and s_{ij} represents coefficient of ith basis function to jth image. S is the non-linear function which poses the higher cost when activities are spread over many basis functions (i.e. almost all coefficient have non-zero value). We choose $log\left(1+s_{ij}^2\right)$ for $S(\cdot)$, following [5]. λ and σ are positive constants that determine the contribution of second term in the cost function (eq.1) and that scales the coefficients, respectively. Sparseness of the coding depends on values of λ and σ. To investigate whether sparseness is crucial for the construction of curvature selectivity in V4, we varied λ and σ systematically so as to alter sparseness of the coding. λ and σ were chosen from the following set:

$$\lambda = \{2.2 * 10^x | x = -2, -1, 0, 1, 2\}, \tag{2}$$

$$\sigma = \{0.316 * 10^x | x = -2, -1, 0, 1, 2\}\ . \tag{3}$$

The values of $\lambda = 2.2$ and $\sigma = 0.316$ were used in Olshausen's report [5]. Those values were altered by a factor of ten. We used all 25 combinations of λ and σ, and calculated sparseness of the coding to investigate quantitatively the relationship between sparseness and the characteristics of computed basis functions. Sparseness of coefficient vector $\boldsymbol{s} = \{s_{ij}\}$ was defined in the same way as the physiological study [8]:

$$\text{sparseness}(\boldsymbol{s}) = (1 - RD)/(1 - 1/n), \tag{4}$$

$$\text{RD} = \frac{1}{n}\left(\left(\sum|s_{ij}|\right)^2 / \sum s_{ij}^2\right) \tag{5}$$

, where n is the number of basis functions (n = 64 in the present study). Sparseness is ranged from 0 to 1; sparseness = 1 means that input images are described by a small number of basis functions (i.e. represented by sparse code).

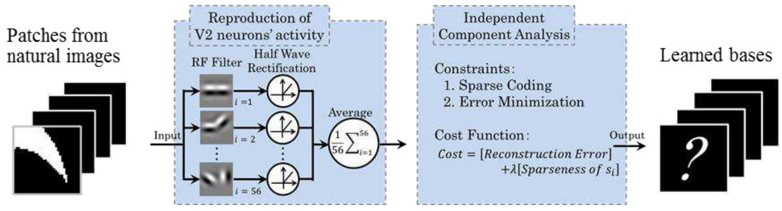

Patches from natural images
Reproduction of V2 neurons' activity
RF Filter
Half Wave Rectification
i = 1
i = 2
i = 56
Input
Average
$\frac{1}{56}\sum_{i=1}^{56}$
Independent Component Analysis
Constraints:
1. Sparse Coding
2. Error Minimization
Cost Function:
Cost = [Reconstruction Error] +λ[Sparseness of s_i]
Output
Learned bases
?

Fig. 1. Schematic illustration of processing flow of the model. Component analysis with sparseness constraint was applied to activities of model V2 neurons that respond to natural images. The receptive fields of model V2 neurons consist of combination of two Gabor filters. The learned bases were predicted to show the receptive field properties of V4 neurons.

2.3 Evaluation of the Basis Functions

To evaluate the selectivity of each basis function, we computed activity map in curvature/direction domain in a way similar to the physiological study reported by Carlson et al. [3] except for the cancelation of stimulus bias in population analysis. The stimulus set used for the evaluation was comprised of a variety of curvature and its direction as examples shown in Fig. 2a. It was defined that a stimulus included a number of distinct curvatures and their directions along its contour. The activity of a basis function to a stimulus was computed by a convolution of the basis and the stimulus. We plotted the activity to the map of curvature and its direction. The activity of a basis to a stimulus was stored into the bins corresponding to the curvatures and directions of all points along the stimulus contour. This procedure was repeated for all stimuli, and the activities were summed for each bin, and divided by the sample number of the bin to obtain a mean. Subsequently, the means were normalized so that they range between 0 and 1.

To evaluate the population preference of 64 bases, we computed a mean of the 64 maps for a combination of λ and σ. Here, it should be noted that the stimulus set for evaluation yields a bias in the population preference. The bias is caused by the facts that the structure of bases was localized around the center of the patch, and that acute curvature was localized around the center. Because of the localization of the basis structure, though stimulus contours around the center were evaluated, those in periphery were not taken into account for the evaluation. However, such localization is not considered for the averaging (we simply divided the summed activity by the sample number for each bin), producing the bias toward acute curvatures that are localized around the center. Since the physiological study [3] reported a bias toward acute curvature, we have cautiously considered this bias. Fig. 2b shows the sample number of each bin in which we observe frequent samples in obtuse curvatures (around 0), although they barely evoke response. To cancel out this bias, we subtracted baseline activity from the population activity. The baseline was the mean activity of basis functions that were learned by randomizing activities of model V2 neurons. After the subtraction of the baseline, the population activity was normalized so that their values range from 0 to 1. We chose this procedure rather than using random bases because a

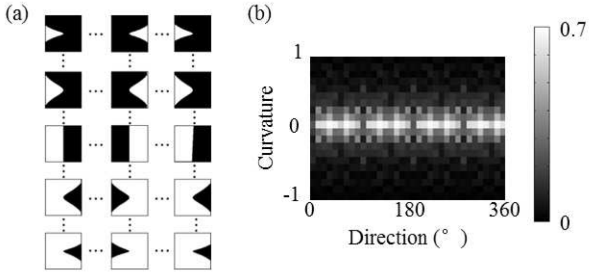


Fig. 2. (a) Test stimuli used for computation of selectivity. Each of them had different curvature and its direction. (b) Distribution of frequency of curvature and its direction in the test stimuli. Obtuse curvature tended to be sampled frequently.

simple randomization appears to induce another bias due to the digitization within a small patch of 33x33 pixels.

3 Results

We propose that the sparse representation is crucial for the construction of curvature selectivity in V4. To test the proposal, we carried out the simulations of the model, and compared the characteristics of single basis functions and their population with those of V4 neurons [3].

3.1 Characteristics of Single Basis Functions

The computed basis functions were dependent on their sparseness (the range of realized sparseness was between 0.38 and 0.88). Fig. 3 showed examples of the basis

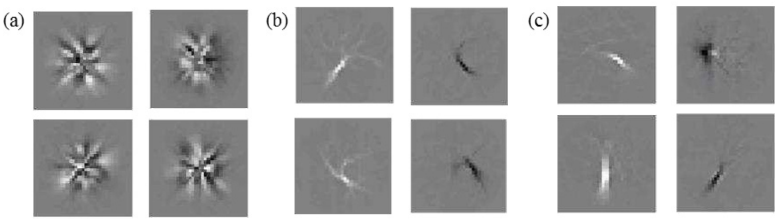


Fig. 3. Examples of basis functions learned with different sparseness. (a) Sparseness was 0.38. No apparent structure was found in the bases. (b, c) Sparseness was 0.81 and 0.87, respectively. They had localized structures that appear to represent the curvature and direction.

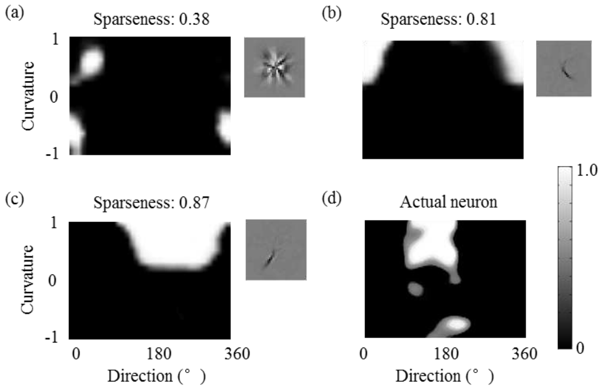


Fig. 4. Selectivity of the single basis function and actual V4 neuron were plotted in curvature and direction domain. (a-c) Tuning function of one basis function from Fig. 3a, b, and c, respectively. Small icon on right of tuning function indicated the basis function of which tuning was computed. Sparseness of the code was 0.38 (a), 0.81 (b), and 0.87(c). (d) Selectivity of actual V4 neuron reported by Carlson et al [3]. Single basis function showed selectivity for curvature and its direction as similar to the physiology when sparseness was medium to high (b, c).

functions learned with different sparseness. Structures of bases seemed to be not organized when sparseness was low (0.38; Fig. 3a). In contrast, there were localized structures that appear to represent the curvature and its direction, when sparseness was medium to large (> 0.6; 0.81 for Fig. 3b, and 0.87 for Fig. 3c). To evaluate whether the bases represent the curvature and its direction, we calculated selectivity of each basis in curvature and direction domain. The tuning functions of one example basis from Fig. 3a-c were plotted in Fig. 4a-c, respectively. Fig. 4d showed the selectivity of actual V4 neuron reported by Carlson et al. [3] in which clear tuning for curvature and its direction. A basis function showed no systematic selectivity for curvature and its direction when sparseness was low (Fig. 4a). However, the bases showed tuning for curvature and its direction when sparseness was medium to large (Fig. 4b, c). These results indicate that sparse representation is crucial for the construction of curvature selectivity in single V4 cells.

3.2 Characteristics of Population

We demonstrated that the basis functions were selective to curvature and its direction when sparseness was medium to large (>0.6). In this section, we examined whether population response of the bases reproduces the physiological characteristics. Carlson et al. have reported that the population activity of V4 neurons showed a bias toward acute curvature with uniform distribution along the direction ([3]; Fig. 5d). Fig. 5a, b, and c show the population selectivity of 64 basis functions whose sparseness were 0.38, 0.81, and 0.87, respectively. The population activity showed the bias toward acute curvature when sparseness was medium (0.6-0.8; e.g. 0.81 for Fig. 5b). In contrast, the population activity did not reproduce the physiological result when sparseness was not medium (Fig. 5a, c). These results indicate that middle to large sparseness is crucial for the construction of the population characteristics of V4

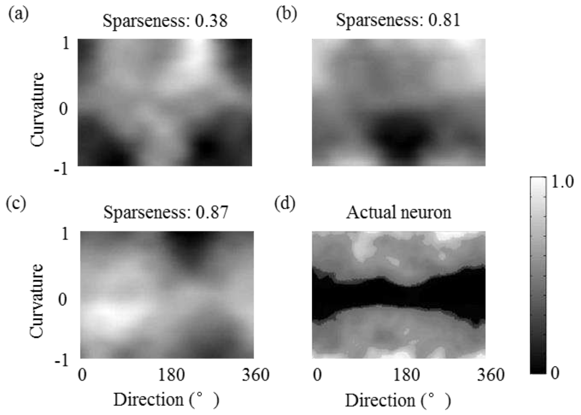


Fig. 5. Population selectivity of the bases (a-c) and the physiology (d; [3]). (a-c) Population activities of the bases mentioned in Fig. 4a-c were plotted. Population activity showed the bias toward acute curvature only when sparseness was medium (0.81; b), corresponding to the physiological study.

neurons. Combining the analyses of single basis and population, we are able to conclude that a sole control of sparseness generates the curvature selectivity reported in V4, including tuning of single cells and characteristics in population.

4 Discussion

In the present study, we proposed that sparse representation is crucial for curvature selectivity in V4. To test the proposal, we applied component analysis with sparseness constraint to the activities of model V2 cells, and investigated whether the characteristics of computed bases correspond to those of actual V4 neurons. The appearance of the basis functions were differed depending on their sparseness. They had localized structures that appear to represent curvature and direction when sparseness is medium to large (>0.6; Fig. 3b, c). To investigate whether these basis functions truly represent the curvature and its direction, we computed the selectivity of each basis functions. The selectivity of single basis functions reproduced the physiological characteristics in terms of tuning for curvature and its direction, when sparseness is medium to large (> 0.6; Fig. 4b, c). Furthermore, we investigated whether population preference of the bases showed the bias toward acute curvature. The population activity reproduced the bias when sparseness was medium (0.6-0.8; Fig. 5b). These results indicate that sparse representation is crucial for curvature selectivity in V4.

We compared the characteristics of the model with those of the actual neurons. Although quantitative comparison of sparseness between the model and the physiological study would provide validity of our proposal, we are unable to compare the sparseness directly because sparseness of V4 cells has not been reported with the stimuli same as our simulations. A physiological study using different stimuli has reported that distribution of sparseness of V4 neurons shows a peak around 0.4 [9]. Although this peak sparseness is slightly small in comparison with that of the present study, having a peak in middle range with symmetric tails appear consistent with our results.

It should be noted that the establishment of the curvature selectivity observed in V4 is not straightforward, e.g., it is difficult to design such connections by hand. Our result is surprising in that the sparseness constrain was capable of generating the curvature selectivity without other heuristic constraints. The implicit constrain in addition to the sparseness was the spatial structure (angle selectivity) inherent in the V2 model that feeds inputs to the V4 model. When we destroyed the spatial structure (angle selectivity) in V2, the learned basis functions did not reproduce the curvature selectivity observed in V4. These results suggest that the curvature selectivity is a natural consequence of the sparse coding of natural images that passed through the visual pathway up to V2.

Sparse representation has potential to explain selectivity of neurons in other cortical regions not limited to the visual cortices because sparse coding is observed physiologically in other cortical regions and modalities (e.g., auditory cortex [10], olfactory system [11], and hippocampus [12]). Theoretical studies have suggested that temporal patterns embedded in sparse input trains are transmitted throughout cortical layers via feedforward connection together with synchronous firing (e.g. [13]). Our study was focused mainly on rate-base neural coding (i.e. coefficients of the basis functions correspond to activities of single neurons). It is expected that further studies on sparse representation give insight in the principle of coding schemes in the cortices.

Acknowledgements. This work was supported by grant-in aids from JSPS (KAKENHI 22300090, 24.3368), and grant-in aids for Scientific Research on Innovative Areas "Shitsukan" (No.25135704) from MEXT, Japan.

References

1. Hubel, D.H., Wiesel, T.N.: Receptive fields and functional architecture of monkey striate cortex. J. Physiol. 195, 215–243 (1968)
2. Ito, M., Komatsu, H.: Representation of angles embedded within contour stimuli in area V2 of macaque monkeys. J. Neuroscience 24(13), 3313–3324 (2004)
3. Carlson, E.T., Rasquinha, R.J., Zhang, K., Connor, C.E.: A sparse object coding scheme in area V4. Curr. Biol. 21, 288–293 (2011)
4. Yamane, Y., Carlson, E.T., Bowman, K.C., Wang, Z., Connor, C.E.: A neural code for three-dimensional object shape in macaque inferotemporal cortex. Nat. Neurosci. 11(11), 1352–1360 (2008)
5. 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)
6. Felleman, D.J., Van Essen, D.C.: Distributed Hierarchical Processing in the Primate Cerebral Cortex, Cereb. Cortex 1, 1–47 (1991)
7. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.W.M.: Amsterdam Library of Objects Images. Int. J. Comp. Vis. 61(1), 103–112 (2005)
8. Vinje, W.E., Gallant, J.L.: Natural stimulation of the nonclassical receptive field increases information transmission efficiency in V1. Science 287, 1273–1276 (2000)
9. Rust, N.C., DiCarlo, J.J.: Balanced increases in selectivity and tolerance produce constant sparseness along the ventral visual stream. J. Neurosci. 32(30), 10170–10182 (2012)
10. DeWeese, M.R., Wehr, M., Zador, A.M.: Binary spiking in auditory cortex. J. Neurosci. 23(21), 7940–7949 (2003)
11. Jortner, R.A., Farivar, S.S., Laurent, G.: A simple connectivity scheme for sparse coding in olfactory system. J. Neurosci. 27(7), 1659–1669 (2007)
12. Quiroga, R.Q., Reddy, L., Kreiman, G., Koch, C., Fried, I.: Invariant visual representation by single neuron in the human brain. Nature 435, 1102–1107 (2005)
13. Asai, Y., Villa, A.E.P.: Integration and transmission of distributed deterministic neural activity in feed-forward networks. Brain Res. 1434, 17–33 (2012)

Handwritten Digit Recognition with Pattern Transformations and Neural Network Averaging

Juan Manuel Alonso-Weber, M. Paz Sesmero,
German Gutierrez, Agapito Ledezma, and Araceli Sanchis

Computer Science and Engineering Department
Universidad Carlos III de Madrid
Avenida de la Universidad 30 Leganés 28911, Madrid (Spain)
jmaw@ia.uc3m.es, {msesmero,ggutierr,ledezma,masm}@inf.uc3m.es

Abstract. Recently there has been a considerable improvement in applications related with isolated handwritten digit and letter recognition supported on the use of deep and convolutional neural networks and other combinations which make use of ensemble averaging. The proposal of the present work is based on a relatively modest sized Neural Network trained with standard Back Propagation and combined with a set of input pattern transformations. Applying ensemble averaging on the trained Neural Networks gives an encouraging error rate of *0.34%* measured on the MNIST dataset.

Keywords: Artificial Neural Networks, Back Propagation, Ensembles, MNIST, Handwritten Digit Recognition.

1 Introduction

Some recent works on Neural Networks applied to handwritten character recognition show an interesting improvement driven by the use of new neural models. Convolutional Neural Networks and Deep Neural Networks are both rather complex structures, which allow reaching a highly respectable performance measured on the popular MNIST Database [1][2]: 0.4% [3] and 0.35% [4]. Combining these architectures with committees or integrating them with ensemble-like structures allows to further improve down to 0.27% [5] or even 0.23% [6]. Using committees with a traditional MLP displays an error rate of 0.39% [7]. Other interesting works which are based on different approaches reach an error rate of 0.40% [8] and 0.32% [9] respectively.

The present work, derived from [10][11], shows a promising approach which advocates for the use of the standard Back Propagation learning algorithm with a relatively modest sized Multilayer Perceptron (MLP). A specific alternative pattern deformation is combined with other usual transformations (displacement and rotation), with an input size reduction and an additive input noise schedule. This helps to avoid local minima and stalling during the learning process.

© Springer-Verlag Berlin Heidelberg 2013

The outcome is a creditable mean error rate of 0.46% tested on the MNIST Database. Applying ensemble averaging on a collection of trained Neural Networks helps to reduce the final error rate down to 0.34%.

2 Data Processing

The MNIST Database contains 70000 digitized handwritten numerals distributed in ten different classes. The whole dataset is divided into 60000 images for training purposes, and the remaining 10000 are reserved for the test set. The graylevel values of each pixel are coded in this work in the [0,1] interval, using a 0 value for white pixels and 1 for black ones.

An important point for managing a high performance in the learning process is the construction of a useful training set. The 60000 different patterns contained in the MNIST database can be seen as a rather generous set, but evidence shows that the usual learning algorithms run into serious trouble for about one hundred or more of the test set samples [10]. Therefore, some strategy is needed in order to increase the training set cardinality and variability. Usual actions comprise geometric transformations such as displacements, rotation, scaling and other distortions. Here a specific set of transformations is combined with an image size reduction and an additive input noise schedule. In this work displacements and rotations are combined with an alternative deformation procedure that yields rather good results.

A problem with which the Back Propagation algorithm tackles is the relative high input dimensionality for the original 28x28 sized digits. Using downsized images helps to reduce the error rate in a small amount. Therefore, a second version of both the training and test sets are generated where each pattern is downsized through interpolation to 20x20 pixels.

Each digit is randomly shifted zero, one or two pixels both in the horizontal and in the vertical axis. The final performance is rather sensible to the probability distribution of the different displacements. Finding an *optimal* probability distribution is a cumbersome task. An interesting possibility is to design different displacement schemas in order to reduce the error correlation of the trained networks, which in turn can induce an improvement with the ensemble averaging procedure. This is shown further on in the Experimental Results Section.

The most important transformation relies on the so called deformation, which involves pulling or pushing each of the four image corner pixels in a random amount along the vertical and horizontal axis. The rest of the pixels are proportionally displaced simulating an elastic behaviour. This leads to a combination of partial stretching and/or compression of the image. Fig. 1 illustrates this process. For the full sized images the displacement interval of the corner pixels is [-5, +5] (distance measured as pixels). For the 20x20 sized images, the best results are achieved with displacements in the order of [-4, +4] pixels. In parallel with the deformation, a rotation is applied around the image center selecting a random angle between -0.15 and +0.15 radians. For technical reasons, the deformation and the rotation need to be computed in an inverse way.

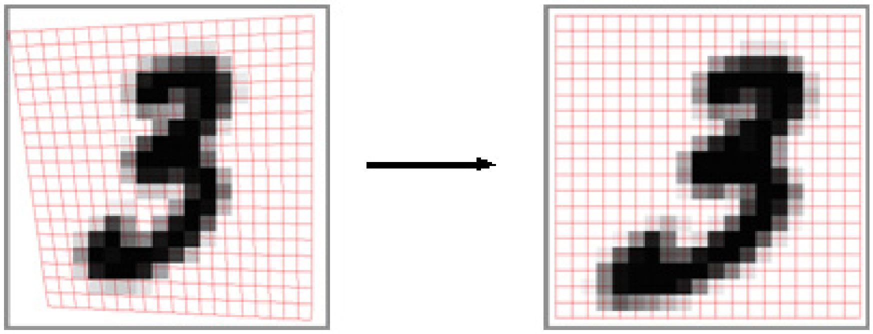


Fig. 1. The deformation is shown as an inverse mapping, at the left the original digit, at the right the deformed one

Finally, noise is added to each image previous to the training process. Noise injection has been extensively related with other techniques such as *weight decay*, and is known to provide a better generalization under certain circumstances [12][13]. Here, best results are obtained with a specific variant of input noise addition that uses annealing: starting with high noise rates that are lowered at a small amount after each learning cycle, and ending with a zero noise rate.

Including the descending input noise schedule improves the MLP precision, on behalf of a longer learning process. A noiseless training requires about 500 cycles, whereas adding the input noise extends the learning up to 1000 cycles. In this circumstance, convergence is tempered down towards the end of the noise schedule. Usually few improvements are achieved during the 100-200 last cycles, and virtually none after the noise scheme is extinguished. The relation between these parameters is $R=N_0/T_{max}$, where T_{max} stands for the number of training cycles, R is the noise reduction value and N_0 is the initial noise value. Using an initial noise value of $N_0=1.0$ adds to each pixel a uniform random value from the interval [-1, +1].

The noise addition and the geometrical transformations are applied to each pattern for each Back Propagation training cycle. Hence, the MLP sees each pattern just once.

The construction of an ensemble with a set of trained Neural Networks is performed with the averaging of the networks outputs which is a simple but effective procedure [14]. For a better performance, it is desirable that the errors committed by the MLPs should be uncorrelated, i.e. the networks should be at most *precise* and *diverse* [15]. Inducing diversity through some random process, for example, training networks with different weight initialization is a correct procedure [14] but has a limited effectiveness [16]. A higher diversity can be accomplished through a guided design of the ensemble [17][18] or inducing additional differentiation in the training process [16], for example modifying the input space. Here both approximations are tried out: a) with a ranked selection methodology, and b) with a set of displacement schemas.

3 Experimental Setup

All the experimentation is built up around training at first a collection of Multilayer Perceptrons, which are afterwards used to apply and evaluate the proposed ensemble averaging.

The MLPs have a fixed size by default: *784×300×200×10*, for the full sized image database. The output class is coded in the usual way as a one-out-of-n schema. For the downsized *20×20* images the only change is for the input layer, which needs 400 input units. The training process is performed using online Back Propagation, where all patterns are presented to the input, structured in cycles. In this case, each pattern is processed previously, applying the above mentioned geometrical transformations combined with the noise addition. The weight initialization is restricted to the [-0.3, +0.3] interval, and the Learning Rate is set to 0.03.

A subset of the training patterns (10000 out of 60000) is randomly removed for validation purposes. This validation set can be used in several ways during the neural network training facing the posterior ensemble averaging: at first, for determining the stopping point of the learning process, and secondly as a criterion for establishing a ranking inside a set of trained neural networks.

As already stated, including the descending input noise schedule improves the final MLP precision, at the cost of a longer lasting learning process. As a rule, the annealing scheme lasts $T_{max}=1000$ cycles, and 100 noiseless cycles are added at the final stage in order to ensure a stable convergence. The initial noise value is $N_0=1.0$, and the descending noise rate $R=0.001$.

The training process of the MLPs was performed on Intel Core i7 and equivalent Xeon processors. Each processor allows to train up to 8 MLPs in parallel without a noticeable loss of performance. Given the MLP size, the size of training set and the needed cycles, the whole learning process lasts about 20 hours for the 20x20 images, and 24 hours for the full sized images.

4 Experimental Results

This section presents the experiments performed at first with the full sized images, and then with the downsized images that achieve a slightly better performance.

At first, a set of 90 MLPs are trained with 50000 images from the MNIST database, leaving 10000 randomly chosen digits for validation. Applying ensemble averaging on the whole MLP set gives an error rate of 0.39%. The drawback of this procedure is that adding or deleting members of the ensemble can vary the results within a range. Leaving out one member selected at random varies the final output between 0.38% and 0.41%.

Following the idea behind the statement that "*many could be better than all*" [17][18], a methodology for selecting then MLPs for the averaging procedure is proposed: validation errors are used in order to establish a ranking for the trained MLPs. The 20 best ranked MLPs are distributed into four subsets named *p, q, r* and *s*, where *p* contains the five neural networks that perform best on the validation set, and *s* the worst. Several ensembles are then built starting with the best subset (*p*), and progressively adding the subsets *q*, *r* and *s*. Table 1 shows the Test Errors committed for these ensembles. Also shown is the evolution of the Mean Test Error for the selected MLPs, which is lower for the better ranked ones. Experiments performed on various MLP sets with different cardinalities suggest that using different seeds for weight initialization derives in a limited differentiation, i.e. the individual MLPs have highly correlated errors. The ensemble averaging tends to acquire the best performance with nine to fifteen members.

In order to establish a reference for evaluating the heuristic, another procedure for building ensembles is included: four *K*-sized ensembles (*K*=5, 10, 15, 20) whose members are randomly selected on 1000 trials from the whole MLP set (i. e. 90 members). This allows establishing a mean value for reference. The results in Table 2 show that this value converges steadily to 0.39%. The ranked selection methodology gives a slightly better performance at 0.36%.

Table 1. Test Errors (in %) for the ensembles built with the progressive addition of the MLPs with the best validation values

		Ranked MLPs				**All**
	(sub)set:	**P**	**p,q**	**p,q,r**	**p,q,r,s**	**90**
MLPs	Mean	0.488	0.485	0.495	0.492	0.51
	Min	0.46	0.45	0.45	0.45	0.43
	Max	0.52	0.53	0.61	0.61	0.61
Ensembles		0.38	**0.36**	**0.36**	0.40	0.39

Table 2. Test Errors (in %) for 1000 ensembles built with *K* randomly selected MLPs

		Randomly selected MLPs				**All**
	***K* =**	**5**	**10**	**15**	**20**	**90**
Ensembles	Mean	0.406	0.397	0.394	0.393	0.39
	Min	0.33	0.33	0.33	0.33	-
	Max	0.49	0.47	0.45	0.45	-

Whereas the MLPs trained on the original MNIST database achieve a mean error rate of 0.51%, using the *20×20* downsized images allows for a slightly lower 0.46%. The following experiment shows that using these MLPs for building ensembles indeed improves the results.

Although the use of the continuous random deformations generates a training set with a virtually unlimited number of patterns, in practice, leaving out a fraction of the original pattern set for validation purposes leads to a descent in performance, especially with the *20x20* sized image set. Therefore, in the following experiments the whole original training set without any geometrical transformations is used for validation purposes. The drawback is that this *particular* validation set does not allow using the ranked selection method on the trained MLPs: the number of validation errors seems to have low relation with the behaviour of the MLPs on the test set in the averaging procedure.

In order to increase the diversity between the trained MLPs, different displacement schemas are established: each image is shifted along the vertical and horizontal axis combined with the remaining deformations, prior to the training process. Each schema determines a different probability distribution of the possible displacements (shown in Figure 2 and Table 3).

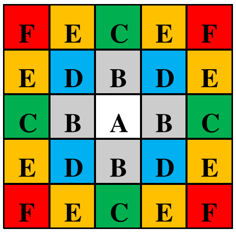


Fig. 2. Different displacement schemas. All the image pixels are displaced the same distance and in the same direction from position A to position types B/C/D/E/F with a certain probability (shown in Table 3). Destination A implies no displacement.

Table 3. Probability distribution of the displacement schemas

	Displacement Schema					
Dest.	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
A	0.43	0.27	0.22	0.20	0.16	0.11
B	0.57	0.49	0.45	0.32	0.41	0.30
C	-	0.16	0.22	0.32	0.05	0.15
D	-	0.08	0.11	0.16	0.31	0.20
E	-	-	-	-	0.08	0.18
F	-	-	-	-	-	0.05

For this experiment 10 Neural Networks for each displacement schema were trained (on *20×20* images). The Mean Test Errors for the six groups varies between 0.465% and 0.496% (shown in Table 4). Averaging each MLP group provides a decrease in the error rates that varies between 0.36% and 0.41%. The full ensemble contains 60 members and displays an error rate of 0.34%. For a more precise result, ensembles are generated with the random deletion of 1 or 2 members of each displacement schema (on 1000 trials), giving a mean error rate of 0.345% and 0.348% respectively.

Table 4. Ensembles built with six MLP sets trained each with a different displacement schema

	Displacement Schema					
	D1	**D2**	**D3**	**D4**	**D5**	**D6**
MLPs	0.487%	0.468%	0.469%	0.465%	0.485%	0.496%
Ensembles	0.39%	0.36%	0.36%	0.36%	0.41%	0.36%
Full Ensemble	**0.34%**					

Averaging all the MLPs trained with the six different displacement schemas shows a lower error rate than those committed by the best ensemble based on an individual displacement schema.

5 Conclusions

This work shows that training traditional Neural Networks with the standard Back Propagation algorithm in combination with an averaging procedure can provide fairly low error rates that lie not too far away from other leading results that rely on rather more complex neural models. The key issues are the generation of an adequate training pattern set, and the use of an additive input noise schedule. For an increased benefit in the ensemble averaging procedure, both a ranked selection methodology and a set of displacement schemas have been presented.

Acknowledgement. This work has been supported by the Spanish Government under projects TRA2011-29454-C03-03 and TRA2011-29454-C03-02.

References

1. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
2. LeCun, Y., Cortes, C.: THE MNIST DATABASE of handwritten digits, `http://yann.lecun.com/exdb/mnist/`
3. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: Seventh International Conference on Document Analysis and Recognition, vol. 1, pp. 958–963 (2003)
4. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, Big, Simple Neural Nets for Handwritten Digit Recogntion. Neural Computation 22(12), 3207–3220 (2010)
5. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional Neural Network Committees for Handwritten Character Classification. In: International Conference on Document Analysis and Recognition, vol. 10, pp. 1135–1139 (2011)
6. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column Deep Neural Networks for Image Classification. In: IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, pp. 3642–3649 (2012)
7. Meier, U., Ciresan, D.C., Gambardella, L.M., Schmidhuber, J.: Better Digit Recognition with a Committee of Simple Neural Nets. In: International Conference on Document Analysis and Recognition, vol. 1, pp. 1250–1254 (2011)
8. Ranzato, A.M., Poultney, C., Chopra, S., LeCun, Y.: Efficient Learning of Sparse Representations with an Energy-Based Model. In: Advances in Neural Information Processing Systems 19, NIPS 2006 (2006)
9. Cruz, R., Cavalcanti, G., Ren, T.: Handwritten digit recognition using multiple feature extraction techniques and classifier ensemble. In: International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 215–218 (2010)

10. Sesmero, M.P., Alonso-Weber, J.M., Gutiérrez, G., Ledezma, A., Sanchis, A.: A new artificial neural network ensemble based on feature selection and class recoding. Neural Computing and Applications 21(4), 771–783 (2010)
11. Alonso-Weber, J.M., Sanchis, A.: A Skeletonizing Reconfigurable Self-Organizing Model: Validation Through Text Recognition. Neural Processing Letters 34(1), 39–58 (2011)
12. Matsuoka, K.: Noise Injection into Inputs in back-propagation learning. IEEE Transactions on Systems, Man, and Cybernetics 22(3), 436–440 (1992)
13. An, G.: The Effects of Adding Noise During Backpropagation Training on a Generalization Performance. Neural Computation 674, 643–674 (1996)
14. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research 11(1), 169–198 (1999)
15. Dietterich, T.: Ensemble methods in machine learning. In: Multiple Classifier Systems, pp. 1–15 (2000)
16. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. Information Fusion 6(1), 5–20 (2005)
17. Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble methods for hybrid neural networks. In: Mammone, R.J. (ed.) Neural Networks for Speech and Image Processing, ch. 10. Chapman-Hall (1993)
18. Sharkey, A.J.C., Sharkey, N.E., Gerecke, U., Chandroth, G.O.: The 'Test and Select' Approach to Ensemble Combination. In: Multiple Classifier Systems, pp. 30–44 (2000)

Echo State Networks in Dynamic Data Clustering

Petia Koprinkova-Hristova and Kiril Alexiev

Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Acad. G. Bonchev str. bl.25A, Sofia 1113, Bulgaria
{pkoprinkova,alexiev}@bas.bg

Abstract. The present paper follows the initial work on multidimensional static data clustering using a novel neural network structure, namely Echo state network (ESN). Here we exploit dynamic nature of these networks for solving of clustering task of a multidimensional dynamic data set. The data used in this investigation are taken from an experimental set-up applied for tasting of visual discrimination of complex dot motions. The proposed here model, although far from complicated brain theories, can serve as a good basis for investigation of humans perception.

Keywords: Echo state networks, dynamic data clustering, visual discrimination of complex motions.

1 Introduction

In spite of numerous developments, clustering of multidimensional data sets is still a challenging task [10]. There are different approaches for solving it that include variety of intelligent techniques as fuzzy logic and neural networks.

Many well known types of recurrent neural networks (RNN) were successfully used for data classification [1, 7, 8]. All of them rely on unsupervised learning procedures minimizing given energy function in search of correspondent to data structure adjustment of network equilibrium states. An extensively developed branch of RNN called “reservoir computing” is targeted mainly to increasing of training speed of these dynamic networks [13]. A representative member of this family is Echo state network (ESN) [9]. It incorporates a randomly generated recurrent reservoir with sigmoid nonlinearities of neurons outputs (usually hyperbolic tangent). There are several works proposing methods for improvement of the ESN reservoir. Most of them are related to entropy maximization [14, 16] and are motivated by known biological mechanisms of changing neural excitability in accordance with the distribution of the input stimuli [15]. In all cases a bias term was used that moves the operating point of the system in the desired direction. In [15] the authors proposed a gradient method named Intrinsic Plasticity (IP) training for adjusting the biases as well as of an additional gain term aimed at achieving the desired distribution of reservoir output. In our previous work [11] it was shown that in fact IP training stabilizes even initially unstable reservoirs. During investigations why and how IP reservoir improvement

© Springer-Verlag Berlin Heidelberg 2013

influences its stability we observed another interesting effect: the reservoir neurons equilibrium points are not only moved but also are concentrated in several regions. Then question aroused: is it possible to use this effect for clustering purposes too?

In [17] for the first time it was proposed to use ESN in image classification to "draw out" silent underlying features of the image data. These extracted features were used further as inputs to a feedforward neural network classifier. In [12] we exploited the same reservoir ability but looking from another perspective: we consider combinations between steady states of each two neurons in the reservoir as numerous two-dimensional projections of the original multidimensional data fed into ESN input. These low dimensional projections can be used next for data clustering. It was shown experimentally that together with improved stability the IP tuned ESN reservoirs possess also better clustering abilities that naturally opens the possibility to apply them for multidimensional data clustering. Based on investigated effect of IP improvement of ESN reservoir we propose a procedure for multidimensional data clustering. It allows discovering multidimensional data structure without specification of the clusters number in advance.

Here we proceed further exploiting the dynamic nature of ESN to reveal their ability to cluster multidimensional dynamic data too. The data used in this investigation are taken from an experimental set-up applied for tasting of visual discrimination of complex dot motions [2, 4]. The proposed here model, although far from complicated brain theories, can serve as a good basis for investigation of humans perception.

2 Problem Formulation

2.1 Echo State Networks for Data Clustering

Figure 1 below represents the basic structure of an ESN as it was used in [12]. The ESN reservoir dynamics is described as follows [9]:

$$r(k) = f^{res}\left(W^{in}u(k) + W^{res}r(k-1)\right) \tag{1}$$

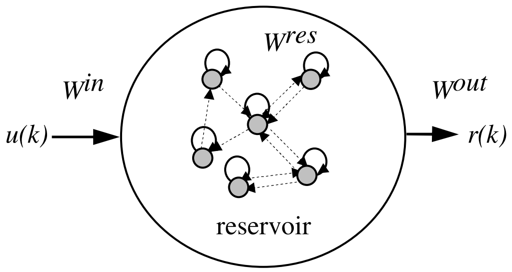


Fig. 1. ESN structure

The IP reservoir improvement proposed in [15, 16] is gradient descent procedure that minimizes the Kullback-Leibler divergence:

$$D_{KL}\left(p(r), p_d(r)\right) = \int p(r)\log\left(\frac{p(r)}{p_d(r)}\right) \tag{2}$$

D_{KL} is a measure for the difference between the actual $p(r)$ and the desired $p_d(r)$ probability distribution of reservoir neurons output r. Since the commonly used transfer function of neurons is the hyperbolic tangent, the proper target distribution that maximizes the information at the output according to [15] is the Gaussian one. In order to achieve those effects two additional reservoir parameters - gain a and bias b (both vectors with n_r size) - are introduced as follows:

$$r(k) = f^{res}\left(diag(a)W^{in}u(k) + diag(a)W^{res}r(k-1) + b\right) \tag{3}$$

The IP training is procedure that adjusts vectors a and b using gradient descent.

The reservoir output after presenting given input $u(k)$ for $k = 0 \div n-1$ is determined by the following recursive calculation:

$$\begin{aligned} r(n) &= f^{res}\left(diag(a)W^{in}u(n-1) + diag(a)W^{res}r(n-1) + b\right) \\ r(n-1) &= f^{res}\left(diag(a)W^{in}u(n-2) + diag(a)W^{res}r(n-2) + b\right) \\ &\dots \\ r(1) &= f^{res}\left(diag(a)W^{in}u(0) + diag(a)W^{res}r(0) + b\right) \end{aligned} \tag{4}$$

In the previous work [12] we exploit the equilibrium state of the reservoir $r(n)$ at the last step achieved by presenting a constant input $u(k) = u_c$ for all $k = 0 \div n-1$. In [12] we showed that this equilibrium will differ for different input vectors and hence its value can be exploited for clustering purposes of static data sets.

Since ESNs are recurrent structures, in case of non-constant input their current state depends on the dynamic behavior of their inputs. Hence we decided to try to exploit this property for clustering of dynamically changing inputs $u(k) = g(k)$ for $k = 0 \div n-1$, where $g(k)$ could be some nonlinear function of k. Obviously the final reservoir state $r(n)$ will depend on the specific characteristics of the time series $g(k)$ presented on its input.

Finally, as it was described in [12], the two dimensional projections of all possible combinations between final states of neurons in the reservoir were subjected to clustering procedure in order to separate input data into number of classes.

2.2 Experimental Set-Up for Dynamic Data Generation

The present data set is received in a study that investigates the changes in visual motion perception in older people (mean age 73.9) [2]. Motion information is important for many everyday tasks like navigation, collision avoidance, figure-ground segregation, three-dimensional shape recovery, etc. These tasks are of vital significance for the survival of the individual. The experiments were focused on the age-related changes in the sensitivity to motion direction of dynamic stimuli. The task of the observers was to indicate whether the mean direction of motion appeared to the left or to the right of the vertical. A fixed proportion of moving elements translated in random directions, while the direction of motion of the rest is taken from uniform distribution centred slightly to the left or to the right from the vertical. In order to perform the task the observers have to integrate the motion direction of multiple elements. However, the task performance also depends on humans' ability to ignore the irrelevant information provided by the noisy elements.

In a typical experiment of this type the observers classify signals presented against an additive background noise in a numerous number of trials. The noise present in the correctly classified trials and in the incorrectly classified trials is analyzed in order to reveal the perceptual "template" used for solving the task e.g. to infer the stimulus features that determine the perceptual decisions. Our stimuli could be described as a signal modulated by noise in the presence of background noise.

To analyze the results of the tests a procedure for objective scenario estimation is realized [2]. In this classical framework of this procedure a trajectory detector estimates and classifies useful statistics for each test. A special measure is proposed for estimation of the temporal characteristics of the random scenario that determines the correctness of observer's decision. Two samples were used – frame statistics and trajectory statistics. These statistics describe in some sense the objective reality i.e. what the subjects see on the monitor. The processing of statistics data with joined data of subjects' answers gives us some knowledge about the elements motion information processing and about the change in the cognitive processes with age.

Each observer took part in 5000 trials, the last 500 of which were the same as the first 500. So, there are 4500 unique trials. The stimuli consisted of 25 frames movie sequences showing 48 moving dots. An example of a frame is given on Figure 2. The fused image of the scenario consisting of all 25 frames is given on Figure 3. The dots moved in a circular aperture with radius of 7.0 deg, positioned in the middle of the computer screen. The stimuli were generated and presented with the help of Psychophysics Toolbox [5], which is used for user controlled scenario visualization with strict time constraints.

Due to the complexity of the problem an initial normalization of the participants in the test is carried out aiming to reach one and the same level of correct responses in participants' answers of about 75%.

In our dynamic series data classification procedure we used all 48 dots trajectories from the scenario generator [2]. Hence for every stimulus of 25 consecutive frames we have 48 time series each consisting of consecutive coordinates for 25 time steps of each dot. From each one of these 48 time series we obtain transformed time series consisting of 24 motion direction angles. Then, following the model of human visual perception from [3, 6], we adopt here the receptive fields of MST neurons as in [3] to preprocess time series of our motion directions data.

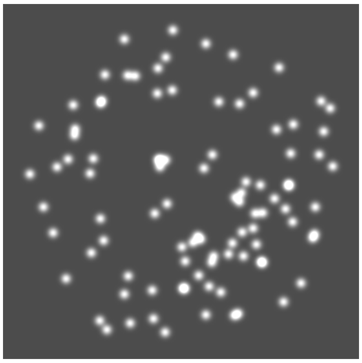


Fig. 2. Random dot motion screen presenting one frame

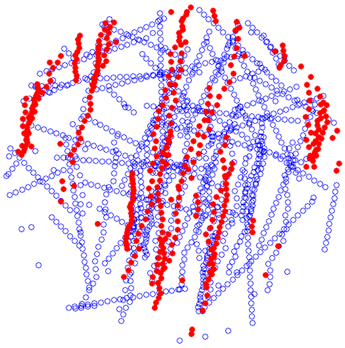


Fig. 3. Fused image of a scenario with 25 consecutive frames

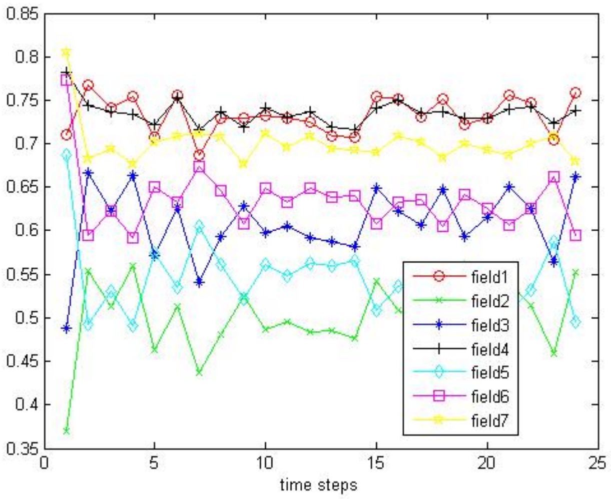


Fig. 4. Receptive fields outputs

We used seven receptive fields distributed randomly in the range of moving angles between –π/2 and +π/2. Finally, at the output of each one of the seven receptive fields we have a time series consisting of 24 time steps for each one of the experiments. Figure 3 represents the time trends of our seven dynamic inputs to the ESN for one of the experiments. These responses are the dynamic 7-dimensional time series data that was used in the role of dynamic input to our ESN clustering procedure.

3 Experimental Results and Discussion

We used 3599 experiments from successive human trails, each containing 25 consecutive screens. After pre-processing of these data using 7 receptive fields we obtain 3599 blocks of dynamical data containing receptive fields' outputs like those on Figure 3, each for 24 time steps. The ESN in this case has 7 inputs. We tested reservoirs with different size starting from 10 neurons up to 100 neurons. The clustering procedure from [12] chooses a two dimensional projection with biggest number of clusters. With increasing of reservoir size we discovered that we are able to discriminate bigger number of clusters. However, in the case of our experimental data, we need to cluster data in two or three clusters since the human decision has three classes: left, right and unclear.

Next we decided to investigate the number of two dimensional projections that cluster the data into two, three, four etc. clusters. In this way an interesting behavior was discovered: the majority of two dimensional projections have only three clusters. Figure 5 presents the bar chart containing the number of projections with different number of clusters for 10, 30, 50 and 100 neurons of ESN reservoir.

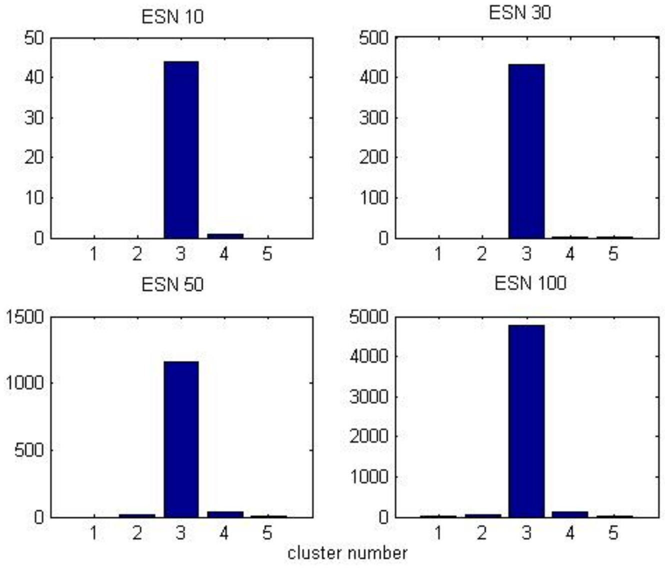


Fig. 5. Number of projections with respective number of clusters

Our main conclusion is that since three clusters are closer to human perception in this the investigated scenario, probably a better idea is to choose projections with prevailing number of clusters. Next we decided to use this as a “voting” mechanism: for each dynamic data block decisions form all two dimensional projections with three clusters were collected and the maximal number of projections that put it into a given cluster was determined; then the data are put to that cluster. In this way we were able to derive dynamic data into three clusters like humans do: right, left and undecided direction of dots movements. Another interesting fact was that approximately 30% of data fall into the “undecided” class like the results obtained in the experiments with human decisions.

4 Conclusions

The investigated here application of ESN to multidimensional dynamic data clustering revealed that the algorithm form [12] works well in the case of dynamic data too. During investigation of the results another way to interpret two dimensional projections was discovered: to use them in a “voting” mechanism and to decide on the basis of majority of similar projections rather than to choose a single projection as it was done in previous work. The developed model reveals also similarity to the decisions to the dot motion discrimination task given by humans. Although this model is far from the complicated brain theories like [3, 6] it can serve as a good basis for investigation of human perception too. Our next step will be to compare proposed dynamical data set clustering procedure with other time series clustering approaches.

Acknowledgments. The research work reported in the paper is partly supported by the project AComIn "Advanced Computing for Innovation", grant 316087, funded by the FP7 Capacity Programme (Research Potential of Convergence Regions).

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science 9, 147–169 (1985)
2. Alexiev, K., Bocheva, N., Stefanov, S.: Assessment of age-related changes in global motion direction discrimination. In: International Conference Automatics and Informatics 2011, Sofia, Bulgaria, October 3-7, pp. B277–B280 (2011)
3. Beardsley, S.A., Ward, R.L., Vaina, L.M.: A neural network model of spiral–planar motion tuning in MSTd. Vision Research 43, 577–595 (2003)
4. Bocheva, N., Bojilov, L.: Neural network model for visual discrimination of complex motion. Comptes rendus de’l Academie bulgare des Sciences 65(10), 1379–1386 (2012)
5. Brainard, D.H.: The Psychophysics Toolbox. Spatial Vision 10, 433–436 (1997)
6. Grossberg, S., Pilly, P.K.: Temporal Dynamics of Decision-Making during Motion Perception in the Visual Cortex, Technical Report BU CAS/CNS TR-2007-001 (February 2008)
7. Hinton, G.E., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)

8. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. of National Academy of Sciences USA 79, 2554–2558 (1982)
9. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology (2002)
10. Jain, A.K., Murty, M.N., Flynn, P.J.: Data Clustering: A Review. ACM Computing Surveys 31(3), 264–323 (1999)
11. Koprinkova-Hristova, P., Palm, G.: ESN intrinsic plasticity versus reservoir stability. In: Honkela, T. (ed.) ICANN 2011, Part I. LNCS, vol. 6791, pp. 69–76. Springer, Heidelberg (2011)
12. Koprinkova-Hristova, P., Tontchev, N.: Echo state networks for multi-dimensional data clustering. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 571–578. Springer, Heidelberg (2012)
13. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review 3, 127–149 (2009)
14. Ozturk, M., Xu, D., Principe, J.: Analysis and design of Echo state networks. Neural Computation 19, 111–138 (2007)
15. Schrauwen, B., Wandermann, M., Verstraeten, D., Steil, J.J., Stroobandt, D.: Improving reservoirs using intrinsic plasticity. Neurocomputing 71, 1159–1171 (2008)
16. Steil, J.J.: Online reservoir adaptation by intrinsic plasticity for back-propagation-decoleration and echo state learning. Neural Networks 20, 353–364 (2007)
17. Woodward, A., Ikegami, T.: A reservoir computing approach to image classification using coupled echo state and back-propagation neural networks. In: Proc. of 26th Int. Conf. on Image and Vision Computing, Auckland, New Zealand, November 29-December 1, pp. 543–458 (2011)

Self-Organization in Parallel Coordinates

Marjan Trutschl*, Phillip C.S.R. Kilgore, and Urška Cvek*

LSU Shreveport, Computer Science Dept.
One University Place
Shreveport, LA, 71115 USA
{mtrutsch,pkilgore,ucvek}@lsus.edu
http://www.lsus.edu

Abstract. Parallel coordinates has shown itself to be a powerful method of exploring and visualizing multidimensional data. However, applying this method to large datasets often introduces clutter, resulting in reduced insight of the data under investigation. We present a new technique that combines the classical parallel coordinates plot with a synthesized dimension that uses topological proximity as an indicator of similarity. We resolve the issue of over-plotting and increase the utility of the widely-used parallel coordinates visualization.

Keywords: Parallel Coordinates, Self-Organizing Map, Visualization, Multidimensional Data.

1 Introduction

A large number of approaches to high-dimensional data span more than a century and a half of work [6,9]. One early example which finds itself in use today is the parallel coordinates plot, a collection of polylines (each representing a single record). Parallel coordinates (PC) has been the subject of protracted discussion in the literature and is accepted as a workable example of high-dimensional visualization. Unfortunately, the nature of parallel coordinates renders it susceptible to over-plotting, especially when the number of records in the data increases. Severe over-plotting results in solid blocks caused by line segments intersecting each other at varying slopes, resulting in diminished utility of the PC plot.

We first present a brief overview of PC, followed by our algorithm for self-organized parallel coordinates (SOPC). We introduce two datasets that showcase its advantages and utilization, and conclude with a summary.

2 Background and Related Work

It is instructive to have a formal definition for the visual components that make up a given PC plot. For the sake of brevity, $\|_{i=1}^{n} x_i$ shall denote the concatenation of all 1-tuples (x_i), or $(x_1, x_2, \dots, x_n)$. Consider a dataset with n variables

* Additionally affiliated with Center for Molecular and Tumor Virology, LSU Health Shreveport (LSUHSC-S), Shreveport, LA 71103 USA.

© Springer-Verlag Berlin Heidelberg 2013

$\mathcal{D} \equiv \|_{i=1}^{n} \mathcal{D}_i$. The plot consists of a list of n axes (such that axis i corresponds to $\mathcal{D}_i$) and a list of polylines. Because the horizontal position x_i at control point i satisfies the condition $x_i \propto i$ (by definition), it is convenient to express the polylines as belonging to a domain $\mathcal{Y} \equiv \|_{i=1}^{n} y_i$, for all $0 \leq y_i \leq 1$. For any given record (or data point) $\boldsymbol{r} \in \mathcal{D}$, a polyline p_r is obtained by con-catenation of the min-max normalization of corresponding components (Eq. 1).

$$p_r \in P \mapsto \Big\|_{i=1}^{n} \left(\frac{r_i - \min D_i}{\max D_i - \min D_i} \right) \tag{1}$$

The shape, or *profile*, of a record can provide an observer with a global perspective of its properties; thus, one may reason that a set of records with similar profiles constitute a single cluster. However, an extremely dense plot impedes the detection of this relationship (Fig. 1). Since a significant portion of records are constrained to a particular region, the profiles of these records become increasingly obscured as density increases, leading to ambiguity.

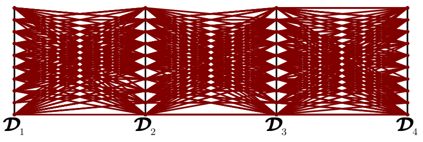


Fig. 1. A dense PC plot can be difficult to interpret, as illustrated here. Many of the records, despite having dramatically different profiles, are difficult to distinguish from one another because of overlap.

Several authors investigated methods of clutter reduction. Dimension reordering is one approach investigated by Peng et al. [13] and XdmvTool [17], and Yang presents a filter and focus+context technique called DOFSA [20]. Polygons represent clusters in the parallel coordinate approach by Novotny [12] and are combined with striped textures. Average shifted histograms visualize density plots with parallel coordinates [11, 19] aiming to alleviate the problem with dimension's bins or frequency intervals. Fua, et al. [7] use hierarchical clustering to generate variable-opacity regions depicting cluster membership.

Extensions of PC into the third dimension (3D) were studied by Wegenkittl et al. [18], visualizing trajectories of high-dimensional dynamic systems and "extruding" PC by creating a third spatial axis and linking parallel planes to extend into 3D. Falkman [4] extends the work and analyzes large amounts of clinical data utilizing parallel planes. Barlow and Stuart [2] use the 3D display to alleviate the problem of coincidence or over-plotting. Parallel glyphs by Fanea, et al. [5] are a 3D extension of PC, changing the axes into star glyphs; we found this to be most similar to our work, with the main difference that star glyphs are formed by "unfolding" the original plot in a radial manner, while ours extrudes the plot into the display medium.

3 Algorithm

We first presented an application of a self-organized parallel coordinates (SOPC) algorithm in [3], and here we present a detailed technique and algorithm study. SOPC is an algorithm based on Kohonen's self-organizing map (SOM) [10] that pulls the records into the third dimension where two or more records can be visually inspected for similarity accross multiple dimensions. Like Kohonen's SOM, the process is characterized by a two-layer feed-forward neural network that utilizes unsupervised learning. A key difference between SOPC and Kohonen's SOM is that the connection between the two layers in SOPC is not a complete graph. We utilize multiple SOM-like grids of output nodes (one per common axis), each associated with one dimension in the dataset. We aim to preserve the presentation of the original PC plot while using a SOM-like approach to address over-plotting/crowding due to high record count while preserving the topology of the original visualization (Fig. 2.b).

3.1 Primary Mapping

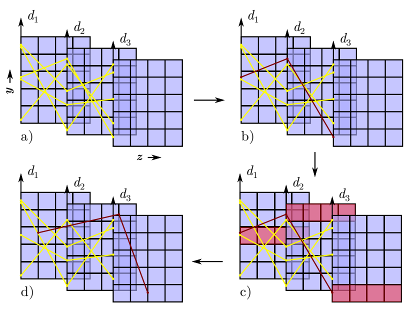
The primary mapping determines the location for an input vector on the common axes (one for each dimension in the dataset), which corresponds to a row of output nodes from space $\mathcal{O}$ in the SOM-like grid (i.e., a self-organizing grid) stretching into 3D. In other words, primary mapping is a function $p'_{\boldsymbol{r}}$ in the domain of primary mapping coordinate vectors (x_p, y_p). Its purpose is to map the record to one of several bins in a $W_p \times H_p$ grid, where each cell is associated with a slice of $\mathcal{O}$. Recall that we previously obtained a list of normalized y values for the record $p_{\boldsymbol{r}}$. These values are scaled by H_p to derive a new list of y' values, while the x values are based on the control point's index i. (Eq. 2)

$$p'_{\boldsymbol{r}} = \mathop{\Big|\Big|}_{i=1}^{n} (i, \lfloor p_{\boldsymbol{r}(i)} H_p \rfloor) \tag{2}$$

The dimensions of the primary grid are particularly important since they direct the granularity of the SOM. Large primary grids are generally preferable to smaller ones when dealing with denser data dimensions; however, they can result in a significantly larger SOM requiring additional system resources. In Fig. 2, we have provided a 4×5 primary mapping.

3.2 Secondary Mapping

Secondary mapping determines a qualifying secondary node for an input vector in the row of output nodes determined through primary mapping. The process is repeated for each dimension. So, for a n-dimensional vector, we perform n secondary mappings. Each node on the self-organizing grid is a collection of random weight vectors with cardinality $||d_w||$ equal to the number of dimensions from the original dataset chosen to be used for mapping to a SOPC plot. Users can



y_1	z_1	y_2	z_2	y_3	z_3	d_1	d_2	d_3
0	3	3	3	0	3	0.1	0.82	0.28
2	1	1	1	2	1	0.5	0.38	0.65
4	1	2	0	3	0	0.9	0.5	0.85
3	0	0	1	4	1	0.88	0.05	0.95
2	3	4	2	0	2	0.53	0.86	0.13

e)

Fig. 2. Basic Principle of a SOPC mapping. a) Primary mapping of input vectors to their respective common axes. b) Mapping of an individual record to primary locations. c) Qualifying secondary output nodes for the selected record. (red) d) Position of the selected record after secondary mapping. e) Data used to construct the SOPC mapping, where each input vector $\boldsymbol{d}$ corresponds to an output node $(\boldsymbol{y}, \boldsymbol{z})$.

determine which dimensions to use for this mapping based on their knowledge and interest in the relationships in the data (similar to the Kohonen SOM). Only the output nodes in the same row as the primary location r are the qualifying secondary nodes for a record (Fig. 2.c). Therefore, the records can organize within a row, instead of the whole grid.

Each output node in the secondary mapping grid is associated with its own n-dimensional weight vector in the `findWinner` phase, which is initially generated randomly such that it conforms to $\mathcal{D}$. We perform this process on each individual control point for a record, which yields an n-list of output nodes representing the record. Using the primary mapping method mentioned above, we select the primary grid cell for control point i as the i^{th} element of $p'_{\boldsymbol{r}}$, and constrain possible *winning nodes* to the subset $\forall z \in [0, D_s], \exists o \in \mathcal{O}_{x,y} : o \in O_{x,y,z}$. We can then generate a set of candidate winning nodes $\mathcal{C}$ as described by Eq. 3.

$$\mathcal{C}_{x,y} \mapsto \forall b \in \mathcal{O}_{x,y}, \exists a \in \mathcal{O}_{x,y} : d(\boldsymbol{a}, \boldsymbol{r}) \leq d(\boldsymbol{b}, \boldsymbol{r}) \tag{3}$$

Because two output nodes in $\mathcal{O}_{x,y}$ may share the same distance from $\boldsymbol{r}$, $\mathcal{C}$ is not a surjection. We solve this issue by applying a lexographical ordering to $\mathcal{C}$ based on the output node's coordinates. Then, finding the winning node for $\boldsymbol{r}$ equals to choosing the infimum of $\mathcal{C}$ in order to determine the winning node. The output for our small example is the dataset in Fig. 2.e. Dimensions $\boldsymbol{r} = (d_1 ... d_n)$ represent an input record; for each $d_i \in \boldsymbol{r}$, a corresponding $(y_i,\ z_i)$ specifies the selected output node, where y_i is an approximation of the position of d_i on the common axis. As the H_p approaches the range of $\mathcal{D}_i$, y_i becomes proportional to d_i.

3.3 Principles of Self-Organization

The `updateNeighborhood` phase permits learning via error correction in a neighborhood about each winning node for a given input vector $\boldsymbol{r}$. In order to improve the accuracy of the mapping, this step is repeated over a number of epochs t_m. During this time, weight vectors within a neighborhood of radius ρ are adjusted based on learning rate λ. The current learning rate λ is a function influenced by three quantities: the initial learning rate λ_i, the final learning rate λ_f, and the current epoch t (Eq. 4). During each epoch, we also calculate ρ from t and ρ_0, the maximum neighborhood radius (Eq. 5).

$$\lambda(t) = \lambda_i \cdot (\lambda_f/\lambda_i)^{(t/t_m)} \qquad (4) \qquad\qquad \rho(t) = \rho_m(1/5\rho_0)^{(t/t_m)} \qquad (5)$$

After these calculations have been performed, the winning node has been found. The output nodes within the neighborhood radius will be adjusted by a factor ω, the neighborhood weight in relation to the winning node (Eq. 6). For these output nodes (x, y, z), new values for each component of its weight vector are derived while updating the neighborhood. For all components $w_{i(x,y,z)}$ in the weight vector $w_{(x,y,z)}$, we perform error correction (Eq. 7).

$$\omega(x, y, z, t) = \lambda(t)e^{\frac{-x^2y^2z^2}{\rho(t)^2}} \qquad (6)$$

$$w_{i(x,y,z)} \leftarrow w_{i(x,y,z)} + \omega(r_i - w_{i(x,y,z)}) \qquad (7)$$

4 Case Studies

While Fig. 2 serves as an example of what SOPC output might look like, it is not terribly useful because it is not dense. Instead, we offer two substantially larger datasets; a synthetic dataset which contains a known number of clusters, and a dataset collected from real-world circumstances.

4.1 Synthetic Dataset

Our synthetic dataset is derived by constructing a dataset with l records randomly distributed from a permutation of b divisions in d dimensions, which yields b^d total clusters. This method was chosen because the dataset it generates is visually noisy and is used as an example of a worse case scenario for a parallel coordinates plot. We created a synthetic dataset using this method with b, d and l set to 4, which yields a 4,096 records with 256 clusters. As one might expect, perceptibility of individual profiles (let alone individual records) suffers significantly (Fig. 3.a).

Even if we visualize and color the data by the cluster assignment dimension (Fig. 3.b), we still have many clusters with varying profiles, and little insight is gained. In order for SOPC to resolve this problem, it must be made apparent that multiple clusters exist and be possible to separate clusters with divergent

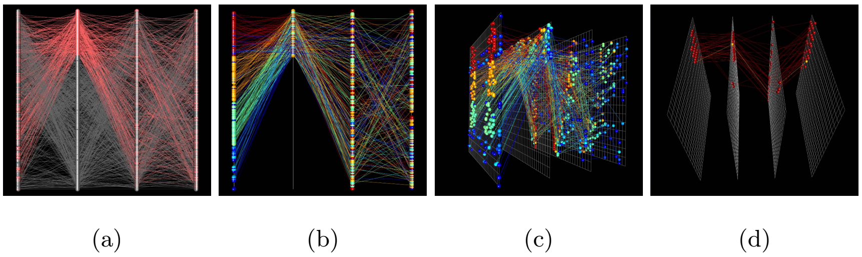


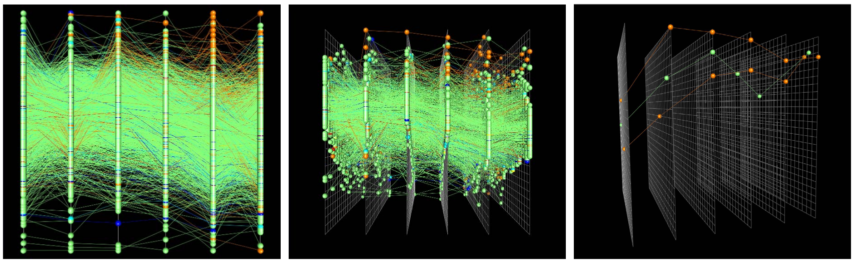
Fig. 3. a) A traditional PC plot exhibits exaggerated clutter. b) The introduction of coloring by original cluster assignment and hiding unselected records does little to resolve perceptual issues associated with (a). c) SOPC performs clutter reduction by assigning records with proximate z values for similar profiles. d) User interaction permits greater exploration of the target data.

properties. As demonstrated by the $4 \times 16 \times 32$ plot in (Fig. 3.c), SOPC permits this through its new, third dimension, which is calculated only using d dimensions (and not the cluster assignment). We can identify several clusters within the data that share similarities in how they were constructed. By interacting with this visualization, its structure can be explored (Fig. 3.d). Upon further investigation of a single "column" in d_0, we can see that clusters represent a permutation of various intervals in each dimension and we can better identify the clusters.

4.2 Jiang-Rhoads Dataset

While the synthetic data gives us the advantage of investigating the worst-case scenario, its evaluation carries some disadvantages with it. We know about cluster count and assignment in the data, but this is rarely the case in real world data. To demonstrate the efficacy of SOPC in this context, we chose a published dataset by Jiang et al. representing microarray gene expression data for 9,298 *C. elegans* (roundworm) genes for each of its six developmental stages [15, 16]. Additional work by Trutschl et al. [16] adds polysomal classes which classify each expression profile by it's polysomal regulation.

When this dataset is visualized using a classic PC plot, pervasive over-plotting is noticeable (Fig. 4.a). Color is mapped to the cluster class variable (Fig. 4.b). One of the first things to note about the resulting visualization is that it highlights outliers, which in this case represent interesting (i.e., atypical) regulatory profiles. For instance, we note a cluster most tightly coupled in the Adult stage (Fig. 4.c), {W07B8.5, Y38H6C.1, F08G5.6}. W07B8.5 encodes cpr-5 (a cysteine protease), and F08G5.6 is involved in defense response [1]. Little is known about Y38H6C.1's ontology, but it is hypothesized to prevent microbial infection, has been shown to be a target gene for DAF-16, and may stimulate germline tumors in *C. elegans* [1,14]. Because of their proximity to one another, hypotheses regarding W07B8.5's and F08G5.6's developmental role with Y38H6C.1 (and perhaps even it's ontology) can be formulated.



(a) (b) (c)

Fig. 4. a) Jiang-Rhoads data as it appears in a parallel coordinates plot, colored by polysomal class. The sheer volume of records makes finding individual record profiles difficult. b) the same data, but presented as a $6 \times 25 \times 25$ SOPC plot. c) A selected cluster from the data which contains a member for which little is known.

5 Conclusion

In this paper we present an extension to the parallel coordinate visualization algorithm. The above test cases demonstrate that SOPC can be used to help resolve ambiguity when presented with an extremely dense dataset. We show that in cases such as these, SOPC may be leveraged to obtain greater insight and clarity into the data. As an interactive visualization, SOPC permits visual exploration that would be difficult in a classical PC plot. We presented two datasets for which SOPC resolves individual clusters in at least two situations. Its application to our synthetic dataset shows that it can deal with situations involving mostly dissimilar record profiles that lead to clutter. Conversely, its application to the Jiang-Rhoads dataset demonstrates its utility for exploratory analysis since it can highlight records whose profiles are outliers.

We are currently investigating further methods of improving perceptibility, such as the dimension reordering approach used by Hurley [8] to provide the best permutations of the coordinates. We have also begun investigating bundling methods via both Bezier curves and Catmull-Rom splines. Finally, we are investigating the usage of force-directed placement to accentuate dissimilar record profiles.

Acknowledgment. Research reported in this publication was supported by an Institutional Development Award (IDeA) and Centers of Biomedical Research Excellence (COBRE) from the National Institute of General Medical Sciences of the National Institutes of Health under grant numbers P20GM103424 and P20GM103433.

References

1. Wormbase web site WS 220 (October 2010), `http://www.wormbase.org`
2. Barlow, N., Stuart, L.: Animator: a tool for the animation of parallel coordinates. In: 8th International Conference on Information Visualization, pp. 725–730 (2004)

3. Cvek, U., Trutschl, M., Stone, R., Syed, Z., Clifford, J., Sabichi, A.: Multidimensional visualization tools for analysis of expression data. World Academy of Sciences, pp. 281–289 (2009)
4. Falkman, G.: Information visualization in clinical ontology: multidimensional analysis and interactive data exploration. A.I. In: Med., 133–158 (2001)
5. Fanea, E., Carpendale, C., Isenberg, T.: An interactive 3d integration of parallel coordinates and star glyphs. In: IEEE Symp. on Info. Vis., pp. 20–27 (2005)
6. Friendly, M., Walker, N.F.: The golden age of statistical graphics. Statistical Science 23(4), 502–535 (2008)
7. Fua, Y., Ward, M.O., Rundensteiner, E.A.: Hierarchical parallel coordinates for exploration of large datasets. In: IEEE Vis., pp. 43–50 (1999)
8. Hurley, C.: Clustering visualizations of multidimensional data. J. Comp. and Graph. Stat., 788–806 (2004)
9. Inselberg, A.: The plane with parallel coordinates. The Visual Computer, 69–92 (1985)
10. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics, 59–69 (1982)
11. Miller, J., Wegman, E.: Construction of line densities for parallel coordinates plots. Computational Statistics and Graphics, 107–123 (1990)
12. Novotny, M.: Visually effective information visualization of large data. In: 8th Central European Seminar on Computer Graphics, pp. 41–48 (2004)
13. Peng, W., Ward, M., Rudensteiner, E.: Clutter reduction in multi-dimensional data visualization using dimension reordering. In: IEEE Symp. on Info. Vis., pp. 89–96 (2004)
14. Pinkston-Gosse, Kenyon, C.: DAF-16/FOXO targets genes that regulate tumor growth in Caenorhabditis elegans. Nature Genetics 39(11), 197–204 (2007)
15. Trutschl, M., Ryu, J., Duke, K., Reinke, V., Kim, S.: Genomewide analysis of developmental and sex-regulated gene expression profiles in caenorhabditis elegans. In: Proceedings of the National Academy of Sciences, pp. 218–223 (2001)
16. Trutschl, M., Dinkova, T.D., Rhoads, R.E.: Application of machine learning and visualization of heterogeneous datasets to uncover relationships between translation and development stage expression of C. elegans mRNAs. Physiological Genomics 21(2), 264–273 (2005)
17. Ward, M.: Xmdvtool: Integrating multiple methods for visualizing multivariate data. In: IEEE Visualization 1994, pp. 326–333 (1994)
18. Wegenkittl, R., Löffelmann, H., Gröller, E.: Visualizing the behavior of higher dimensional dynamic systems. In: 8th Conf. on Vis., pp. 119–125 (1997)
19. Wegman, E.: Hyperdimensional data analysis using parallel coordinates. J. American Stat. Assoc., 664–675 (1990)
20. Yang, J., Peng, W., Ward, M., Rudensteiner, E.: Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In: IEEE Symposium on Information Visualization, pp. 14–21 (2003)

A General Image Representation Scheme and Its Improvement for Image Analysis

Hui Wei[*], Qingsong Zuo, and Bo Lang

Laboratory of Cognitive Model and Algorithm,
Department of Computer Science, Fudan University, Shanghai, China
weihui@fudan.edu.cn

Abstract. In this paper, a bio-inspired neural network is developed to represent images and analysis features of images effectively. This model adopts schemes of retinal ganglion cells (GC) working and GCs' non-classical receptive fields (nCRF) that can dynamically adjust their sizes/scales according to the visual information. Extensive experiments are provided to value the effect of image representing, and experimental results show that this neural network model can represent images at a low cost and with a favor in improving both segmentation and integration processing. Most importantly, the GC-array model provides a basic infrastructure for image semantic extraction.

Keywords: nCRF, Image representation, Multi-scale, Contour detection.

1 Introduction

Efficient representation of visual information is a primary goal of human visual system. The evolution and development of this system can enable human to understand the environment around us quickly and efficiently. So the schemes of the human visual system are studied and researched during the past decades. Efficiency of a representation refers to the ability to capture significant information about an object of interest from scene using a description with minimum cost. From this point, image representation may be designed to rely on the specific constrains and knowledge available for the restricted task at hand. It is widely assumed that a general-purpose visual system must entail multiple types of representation which share data through a variety of computation.

In biological visual systems, the classical receptive field (CRF) of GC is sensitive to brightness and contrast. It demonstrates the attributes that has spatial summation properties enabling the boundaries to be detected [1-4]. The expanded area around CRF, it is referred to as the non-classical receptive field (nCRF), can compensate for the loss of low-frequency information caused by the antagonistic center-surround mechanism of CRF [5]. The receptive field (RF), including CRF and nCRF, is the basic structural and functional unit of visual information processing. Any GC merges

[*] Corresponding author.

© Springer-Verlag Berlin Heidelberg 2013

all stimuli occurring in its RF, and report a composite signal upwards for further processing [6-10]. By means of these dense and regular RFs, a GC-array can produce a general representation of any external stimulus

Many studies have examined the representation of natural images. Previously proposed methods of image representation range from color histograms to feature statistics, from spatial frequency-based to region-based and from color-based to topology-based approaches for an extensive review of image representation techniques, see [6-8]. Another image representation scheme using a set of block pattern models was developed by Yao and colleagues, in which each small block satisfies certain intensity variation constraints [5]. Some studies have attempted to construct models simulate GCs. Li et al [5]. constructed a 3-Gaussian model using it to simulate spatial responses of GCs. Qiu et al.[9] reported a new model of the mechanisms underlying mutual inhibition within disinhibitory nCRFs. A network model is proposed by Qi et al.[10]consisting of neurons and interactive columns, to simulate the integrative tuning curves of the nCRF, and demonstrate the functional roles of the modulations in image processing. Ghosh et al.[11, 12]modeled the nCRF as a combination of 3-Gaussian at three different scales and seeking to explain certain brightness-contrast illusions. Hayashi et al.[13] formulated four topographic attentive mapping networks with a multilayer structure of the nCRF. Kenyon et al.[14] reported that the neural circuity of nCRF can account for the stimulus specificity of high-frequency oscillatory potentials in response to both high and low contrast features. Perrinet et al.[15] reported a model that represents the multi-scale contrast values of images using an orthonormal wavelet transform. In particular, none of these models specified the relationship between the disinhibitory nCRF and neural circuits in the retina and did not take into account of the dynamic adjust nature of the RF of GCs. They also did not propose mechanisms for joining pixels or organizing them fragmentally for clustering an explicit and tentatively assembled representation.

In the computational network, for the purpose of integration, top-down control and the neighborhood properties of input stimuli are taken into account. Our experimental results revealed that RFs become smaller when local areas of images have details that need to be distinguished, and simultaneously they tend to become larger when local areas exhibit no obvious difference. From this point, the RF is considered as a computational unit which is used to locate and represent borders and details of objects.

2 The Design of Computational Model Based on nCRF of GC

We design a model where the response of GC is the convolution of 3-Gaussian function with visual information [16, 17]. Based on the previous work, this section mainly demonstrates the algorithm of computing scales of nCRF. From the neurophysiological perspective, Receptor cells (RC) compose the center area of RF of each bipolar cell (BC). Horizontal cells (HC) integrate the response of RFs, transfer them to BC, inhibit the CRF center of BC, and form the peripheral zone of CRF. In the same way, BCs transfer their response to a GC, and GC's CRF with antagonistic center-surround structure is constructed. Amacrine cells (AC) integrate the response of BC and transfer them to GC and inhibit the CRF surround of the GC, thus forming the GC's nCRF (extra-surroundl). Interplexiform cell (IC) exerts feedback control over HC and BC, such that the cessation of the IC's activity enhances the activity of HC and BC. The number of HC and BC was increased, that form the CRF center and surround of GC, in other words, increasing the size of Gc's RF. conversely, an increase in IC's activity inhibits the activity of HC and BC, such that it decreases the number of HC and BC that form the CRF center and surround of GC, reducing the size of RF of GC. Based on the neurophysiological mechanisms of the GC's RF, the simplified neural circuit shown in Fig.1 was designed to enable the RF to be adjusted automatically and dynamically.

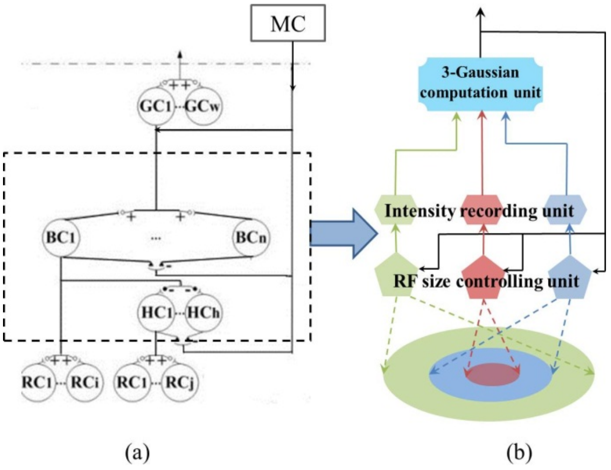


Fig. 1. The computational model based on GC. (a) is a micro neural circuit for RF constitution, where the solid lines represent forward transfer and feedback control, the hollow circle and solid circle represents synapse of facilitation and inhibition, respectively. The simplified computation model based on (a) is shown in (b).

The algorithm of RF adjustment adaptively is designed as follows:

```
1.  At first stage, the photoreceptor cells (RCs)
transform RGB values of pixels to wave length. The
center zone, surround zone and nCRF have their initial
sizes.
2.  All summed RGB information is transmitted upwards,
through different channels of RF size controlling
units.
3. All information was integrated in GC layer (3-
Gaussian computation unit). According to the output of
GC, the upper layer send feedback signal to RF size
controlling units and the RF can dynamically adjust
their size.  The general principles of RF adjust
dynamically.
```

3 Experiments Results

Based on the neural computational model and algorithm described above, we conducted a number of experiments to test and evaluate the efficiency of our model. The aims of these experiments are to test whether a representation using GC-array is implemented, and, if so, whether it could facilitate subsequent advanced image processing, such as segmentation, contour detection and feature extraction. Section 3.1 and 3.2 demonstrate the interesting result.

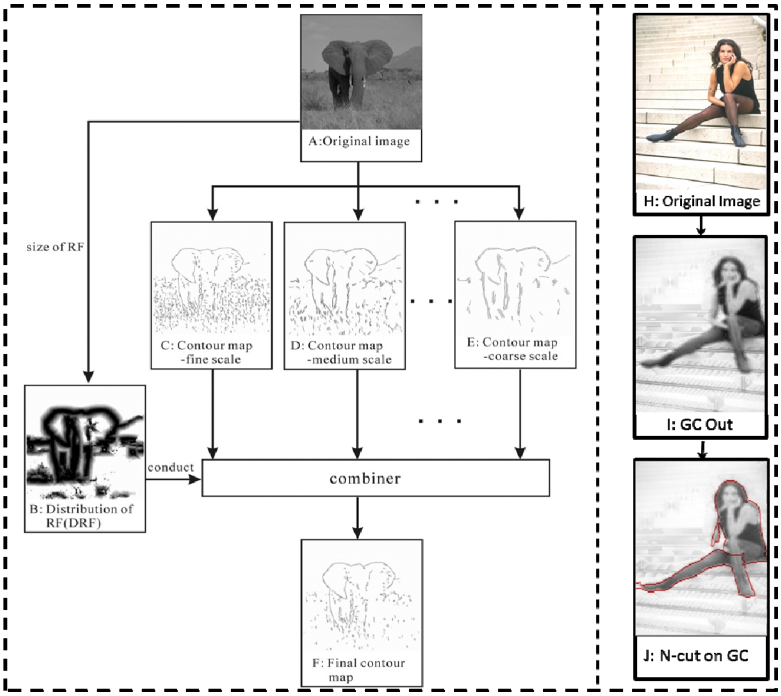

A:Original image
size of RF
C: Contour map -fine scale
D: Contour map -medium scale
E: Contour map -coarse scale
B: Distribution of RF(DRF)
conduct
combiner
F: Final contour map
H: Original Image
I: GC Out
J: N-cut on GC

Fig. 2. Models of experiments. The left part is model of multiscale contour detector based on nCRF. (A) Original image. (B) DRF which is computed by the algorithm—self-adaptive RF in section 2.2. (C) Contour map at fine scale is computed by the algorithm in [20]. (D) Contour map at medium scale. (E) Contour map at coarse scale. (F) Final contour map which is combined by the series of contour maps at different scales. In the right part, the flow diagram shows the experiment of promoting N-cut algorithm. (H) Original image. (I) Output of GC. (J) N-cut result on GC Output.

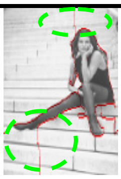
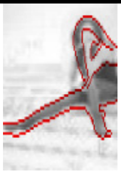
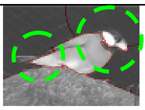
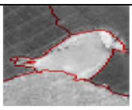
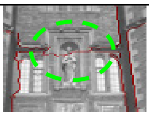
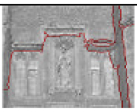
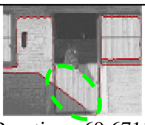
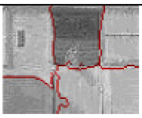
The left part of Fig. 2 shows the schematic drawings illustrating the general flowchart of the proposed model for multi-scale contour detection. In short, given an original image (Fig. 2A), the darker pixels in Fig. 2B representing the finer scale, similarly, the brighter the coarser. Contour detectors [20] with different scales are used to extract a series of contour maps (Fig. 2C, D, E). A series of contour maps can be extracted by the algorithm [20], and the next step is to combine the contour maps, and details can be found in [16]. In the right part of the Fig. 2, GC can output the response to the original image. N-cut algorithm, as an outstanding segmentation method, will produce better result on the GC output than on the original image. These results both illustrate that the GC model based on nCRF is a general and admirable representation scheme.

3.1 Promoting N-cut Algorithm

The N-cut algorithm [18] is a widely used method for image segmentation. We tested the effects of running this algorithm on a GC array instead of pixels. Table 2 shows the results. We applied this algorithm to BMP images and a GC-array, respectively, and then compared the two resulting segmentations. The third column shows that the results were improved relative to the use of the algorithm alone. The green circles mark the specific places in which the segmentation was improved. Importantly, there was a dramatic reduction in the required run-time. Thus, a good GC-based representation can improve the efficiency of segmentation without sacrificing performance.

This advantage of the current method was likely related to the much smaller dimensions of the GC-array compared to the original BMP image, which would be expected to greatly reduce the complexity of the graph-based segmentation algorithms. The computational complexity of the min-cut in time is represented by O(n^2logn), where n is the number of vertices of graph, and a near-optimal min-cut algorithm's computational complexity is O(2nm* log(n^2/m)), where m is the number of edges [19]. Thus, smaller dimensions must reduce the values of n and m.

Table 1. Facilitation of the N-cut algorithm by the GC-array

Original image	**N-cut on pixels**	**N-cut on GC-array**
	 Run time: 34.2777	 Run time: 4.5071
	 Run time: 23.8193	 Run time: 2.8363
	 Run time: 21.3145	 Run time: 3.5533
	 Run time: 69.6711	 Run time: 4.4497

3.2 Promoting the Effects of Contour Detection

Contour detection is considered as a fundamental operation in the image processing and computer vision. The paper [20] proposed a contour detection based on inhibition mechanism of nCRF of neurons in V1. Due to the different size of GCs, we can obtain the different scale information from image. The coarse scale equals to the GC with large size, while fine scale equals to the GC with small size. On this basis, we introduce the preprocessing role of GC in contour detection and extract the object's contour through the combination of multi-scale information. It can significantly prompt the performance of contour detection compared with Grigorescu's method [20].

The evaluation method used was the same as that used by Grigorescu [20]. Fig. 3 shows comparative box-and-whisker plots for five of the images used in our experiments. More details and results can be obtained from [17]. It is clear that, from these plots, our model showed a significantly higher performance in almost all cases.

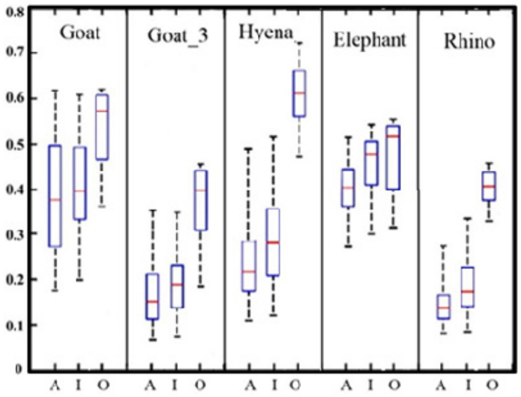
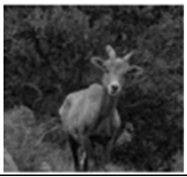
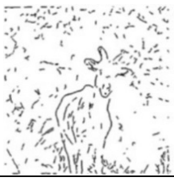
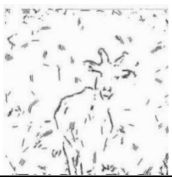
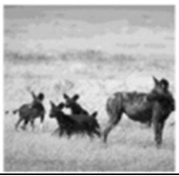
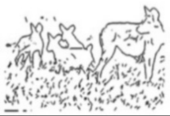
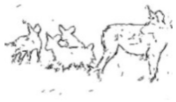
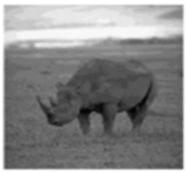
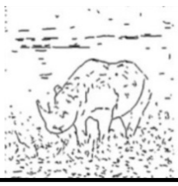
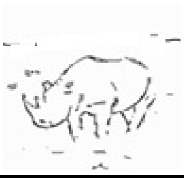


Fig. 3. Contour detection performance comparisons. A: Contour detection performance of the anisotropic model; I: the isotropic model from literature [15]; O: our model. Each lattice represents the performance plots based on a similar image. In each plot, the horizontal red line in the box shows the median value of the performance. The top and bottom lines of each box denote the upper and lower quartiles.

Table 2 shows the results and revealed that the neglectable textures are removed dramatically and the object's contours are extracted effectively at some extent after the preprocessing of GC. In this section, all real images used are downloaded from http://www.cs.rug.nl/~imaging/.

Table 2. GC-array facilitating contour detection

Images	The anisotropy results of [20]	The result on GC-array
		
		
		

4 Conclusion

To support a diversity of higher-level computations, an early visual representation must make important information explicit and discard only redundant or perceptually irrelevant information [21]. Selecting an appropriate representation method for visual stimuli was an essential task for the evolution of sighted animals. Experiments on the applications, including N-cut and contour detection, reveal that this novel representation algorithm plays a very role in their performance. Our representation model is different from other models merely perform segmentation. The properties of GCs' nCRF are adaptive, and nCRF can adjust its filter characteristics according to the spatial frequency components of images. In additional, the nCRF is flexible under conditions of changing contrast and brightness of stimulation. Along with the changing image spatial properties, the nCRF sometimes turns into a high spatial frequency filter and sometimes into a low one.

Overall, recent progress on exploring and researching the image understanding and image retrieving is encouraging. Although, results on these fields can promote practical application in daily life, there is a long way to enable computer to handle and know well complex visual scene. In all probability, the mechanism applied by human in understanding the world around us will serve to guide our thinking in the exploration of higher challenging visual tasks.

References

1. McIlwain, J.T.: Some evidence concerning the physiological basis of the periphery effect in the cat's retina. Exp. Brain Res. 1, 265–271 (1966)
2. Ikeda, H., Wright, M.J.: The outer disinhibitory surround of the retinal ganglion cell receptive field. The Journal of Physiology 226, 511 (1972)
3. Krüger, J., Fischer, B.: Strong periphery effect in cat retinal ganglion cells. Excitatory responses in ON-and OFF-center neurones to single grid displacements. Exp. Brain Res. 18, 316–318 (1973)
4. Feghali, J.G., Jin, J.C., Odom, J.V.: Effect of short-term intraocular pressure elevation on the rabbit electroretinogram. Invest. Ophth. Vis. Sci. 32, 2184–2189 (1991)
5. Shou, T., Wang, W., Yu, H.: Orientation biased extended surround of the receptive field of cat retinal ganglion cells. Neuroscience 98, 207–212 (2000)
6. Fauqueur, J., Boujemaa, N.: Region-based image retrieval: Fast coarse segmentation and fine color description. Journal of Visual Languages & Computing 15, 69–95 (2004)
7. Deng, Y., Manjunath, B.S., Kenney, C., Moore, M.S., Shin, H.: An efficient color representation for image retrieval. IEEE Transactions on Image Processing 10, 140–147 (2001)
8. Saykol, E., Gudukbay, U., Ulusoy, O.: A histogram-based approach for object-based query-by-shape-and-color in image and video databases. Image Vision Comput. 23, 1170–1180 (2005)
9. Chaoyi, Q.F.L.: Mathematical Simulation of Disinhibitory Properties of Concentric Receptive Field. Acta Biophysica Sinica 11, 214–220 (1995)
10. Xianglin, Q., Xiaochuan, P., Yunjiu, W.: A mathematical model of integration field beyond receptive field of cortical neuron. In: Proceedings of the 9th International Conference on Neural Information Processing, ICONIP 2002, vol. 4, pp. 1694–1698 (2002)
11. Ghosh, K., Sarkar, S., Bhaumik, K.: A possible mechanism of zero-crossing detection using the concept of the extended classical receptive field of retinal ganglion cells. Biol. Cybern. 93, 1–5 (2005)
12. Ghosh, K., Sarkar, S., Bhaumik, K.: A possible explanation of the low-level brightness–contrast illusions in the light of an extended classical receptive field model of retinal ganglion cells. Biol. Cybern. 94, 89–96 (2006)
13. Hayashi, I., Maeda, T.: Structure Evaluation of Receptive Field Layer in TAM Network, vol. 2, pp. 1541–1547. IEEE (2006)
14. Kenyon, G.T., Travis, B.J., Theiler, J., George, J.S., Stephens, G.J., Marshak, D.W.: Stimulus-specific oscillations in a retinal model. IEEE Transactions on Neural Networks 15, 1083–1091 (2004)
15. Perrinet, L., Samuelides, M., Thorpe, S.: Coding static natural images using spiking event times: do neurons cooperate? IEEE Transactions on Neural Networks 15, 1164–1175 (2004)
16. Wei, H., Zuo, Q., Lang, B.: Multi-scale image analysis based on non-classical receptive field mechanism. In: Lu, B.-L., Zhang, L., Kwok, J. (eds.) ICONIP 2011, Part III. LNCS, vol. 7064, pp. 601–610. Springer, Heidelberg (2011)
17. Wei, H., Lang, B., Zuo, Q.: Contour detection model with multi-scale integration based on non-classical receptive field. Neurocomputing (2012)
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 888–905 (2000)
19. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiway cuts, pp. 241–251. ACM (1992)
20. Grigorescu, C., Petkov, N., Westenberg, M.A.: Contour detection based on nonclassical receptive field inhibition. IEEE T. Image Process 12, 729–739 (2003)
21. Elder, J.H.: Are edges incomplete? Int. J. Comput Vision 34, 97–122 (1999)

Learning Features for Activity Recognition with Shift-Invariant Sparse Coding

Christian Vollmer[1], Horst-Michael Gross[1], and Julian P. Eggert[2]

[1] Ilmenau University of Technology,
Neuroinformatics and Cognitive Robotics Lab,
98684 Ilmenau, Germany
christian.vollmer@tu-ilmenau.de
[2] Honda Research Institute Europe GmbH
63073 Offenbach/Main, Germany
julian.eggert@honda-ri.de

Abstract. In activity recognition, traditionally, features are chosen heuristically, based on explicit domain knowledge. Typical features are statistical measures, like mean, standard deviation, etc., which are tailored to the application at hand and might not fit in other cases. However, Feature Learning techniques have recently gained attention for building approaches that generalize over different application domains. More conventional approaches, like Principal Component Analysis, and newer ones, like Deep Belief Networks, have been studied so far and yielded significantly better results than traditional techniques. In this paper we study the potential of Shift-invariant Sparse Coding (SISC) as an additional Feature Learning technique for activity recognition. We evaluate the performance on several publicly available activity recognition data sets and show that classification based on features learned by SISC outperforms other previously presented Feature Learning techniques.

Keywords: feature learning, activity recognition, sparse coding.

1 Introduction

Context-aware computing provides intelligent systems with the ability to perceive the world from the user's perspective and allows to provide smart assistance where necessary. Human activity is an important cue for inferring the state and context of a user. In recent years, activity recognition has gained increased attention due to its usefulness and practical success in application domains such as medical diagnosis, rehabilitation, elderly care, assembly-line work, and human behavior modeling in general. As a result, a number of successful approaches have been built for recognition of a wide range of activities.

In most cases, activities are recognized by their body movements, since they are clearly defined by the motion and relative position of the user's body parts. Sensors, attached to the body or embedded into objects that are utilized throughout the activities, are used to capture those movements. One of the main issues

© Springer-Verlag Berlin Heidelberg 2013

in activity recognition is that the sensor readings are typically noisy and often ambiguous. By applying signal processing and pattern recognition techniques, those data can be automatically analyzed, yielding a real-time classification of the activities.

In activity recognition the goal is to detect and classify contiguous portions of sensor data that contain the activities of interest. A widely adopted approach is the sliding window technique, where overlapping frames of the incoming multidimensional signal stream are extracted and a set of features is computed over each frame. The features are then categorized by means of some classifier. Popular features computed from the signal are mean, variance or standard deviation, energy, entropy, correlation between axes or discrete FFT coefficients (see e.g. [2] for a good comprehension). Common methods for classification based on the extracted features include Naive Bayes, Decision Trees, K-Nearest-Neighbors, and Support Vector Machines (see e.g. [6,7]).

As mentioned by Plötz et al. in [7], feature extraction is usually a heuristic process that is driven by domain knowledge about the application at hand. This process has to be repeated for every new application domain and sometimes even for new sensor setups in the same domain. Thus, conventional approaches are usually tailored to specific applications. One way to overcome this restriction and build a general approach is to find methods to automatically discover useful features in the data or, in other words, adapt the features to the data.

Activity recognition techniques are usually built of two main components: (i) a feature extraction technique and (ii) a classifier. Most approaches are using the above mentioned standard features. Only recently, first attempts have been made in applying machine learning techniques to learn features from the data. Plötz et al. [7] use two feature learning techniques, namely PCA and Deep Belief Networks, to automatically discover features. By applying feature learning as a preprocessing step, the authors argue, a universal feature representation is created that captures the core characteristics of the data. The authors show that classification based on the discovered features yields significantly better results, then traditional techniques.

We would like to contribute to that line of research by investigating Shift-invariant Sparse Coding as another technique for feature learning in the area of activity recognition. The idea of Sparse Coding is that the data can be represented as a composition of sparsely distributed features. The data is imagined as consisting of two hidden components, (i) the set of features and (ii) activations of those features that mark, when a feature occurs in the data. The goal is to learn the features as well as their activations from the data in an unsupervised manner. The learning problem can be decomposed into two subproblems, the coding problem and the dictionary learning problem. In the coding problem, the features are assumed to be given and fixed. Here, the goal is to find a minimal set of activations such that the input data is best reconstructed, i.e. the error between input data and linear superposition of the features according to the activations is minimized. In the dictionary learning problem, the activations are assumed to be fixed and the goal is to adapt the features to the data given a known set of activations. By initializing dictionary and activations randomly and alternating the two steps iteratively, one can learn both components simultaneously.

In the domain of time series processing, Sparse Coding has been mainly used for auditory signal coding. In [9], the authors aim at computing a sparse representation of natural audio signals in form of spike trains, where the spikes mark activations of a set of basis functions (or features), which are also learned, and represent an optimal dictionary. The authors argue that such a representation provides a very efficient encoding and uncovers the underlying event-like structure of the signal. More recently, Sparse Coding has been applied to find patterns in movement data, like walking cycles of human legs [4].

We use an approach similar to that published in our earlier work [10], where Sparse Coding has been utilized to learn features from handwriting data and generate handwritten characters using the features and statistics of their typical occurrence. The contribution of this work is the use of that framework for learning features from general activity data, which is much more diverse, and to build a simple classifier using those features.

We compare our results to those published by Plötz et al. [7], where PCA and Deep Belief Networks have been compared to conventional approaches for feature extraction. The authors evaluate their approaches on four publicly available activity recognition data sets. To be comparable, we will evaluate our approach on the same data. The rest of this work is organized as follows. We describe our approach in detail in Sec. 2. In Sec. 3, we compare our approach to previously published approaches on a number of activity recognition data sets. Finally, we discuss our work in Sec. 4 and give a brief outlook on its potential in activity recognition.

2 Method

Feature Learning. We formalize Feature Learning as a Sparse Coding (SC) problem. In general, given an input signal, the goal in SC is to find features (or *basis vectors* in SC terms) and a sparse set of feature occurrences (or *activations* in SC terms) that, when linearly superimposed, reconstruct the input. We use a special kind of SC formulation as a Non-negative Matrix Factorization (NMF) problem. As detailed later, this problem can be solved by minimizing an energy function on the error between reconstruction and input plus a penalty on the activations. By imposing a non-negativity constraint, i.e. basis vectors and activations have to be non-negative, and a sparseness constraint on the activations, the resulting basis vectors are interpretable as features that constitute an alphabet underlying the data [5]. We further use a variant of NMF called Shift-NMF [1]. Shift-NMF introduces translation-invariant basis vectors. Thus, a basis vector can occur anywhere in the input, which is necessary for temporal signals with reoccurring features. In the following, we will give a formal description of the problem and the update equations.

As mentioned earlier, consecutive, overlapping frames are extracted from the input signal before feature extraction. Feature learning is then performed over all frames simultaneously. Let $\mathbf{V}^d \in \mathbb{R}^{N\times T}$ denote the matrix of the N training frames of frame length T, where d indexes the dimensions of the signal. For ease of notation, we separate the dimensions of the signal into distinct matrices, indexed by d. A single frame is denoted as $\mathbf{V}^d_n$ and the scalar elements by $V^d_{n,t}$. Let $\mathbf{W}^d \in \mathbb{R}^{K\times L}$ be the matrix of K basis vectors of length L, with elements $W^d_{k,l}$. We denote the single basis vectors by $\mathbf{W}^d_k$. Let $\mathbf{H} \in \mathbb{R}^{N\times K\times T}$ be the tensor of activations $H_{n,k,t}$ of the k-th basis vector at time t for frame n.

In NMF the input, basis vectors, and activations are constrained to be non-negative. Thus, for NMF to be applicable, the input signal has to be made non-negative. We do this by doubling the number of input dimensions and projecting the negation of its negative parts to the new dimensions. The non-negative input $\tilde{\mathbf{V}}^{\tilde{d}}$ as used in the calculations below is then given by

$$\tilde{\mathbf{V}}^{2d} = \max(\mathbf{V}^d, 0), \quad \tilde{\mathbf{V}}^{2d+1} = \max(-\mathbf{V}^d, 0) \ . \tag{1}$$

For ease of notation, we resubstitute $\tilde{\mathbf{V}}^{\tilde{d}}$ with $\mathbf{V}^d$ again. However, please keep in mind, that $\mathbf{V}^d$ denotes the non-negative input from now on.

We learn $\mathbf{W}^d$ and $\mathbf{H}$ with NMF by minimizing the following energy function

$$F = \frac{1}{2}\sum_d \left\|\mathbf{V}^d - \mathbf{R}^d\right\|_2^2 + \lambda \left\|\mathbf{H}\right\|_1 \ . \tag{2}$$

The matrices $\mathbf{R}^d \in \mathbb{R}^{N\times T}$ are the reconstructions of the frames by activation of the basis vectors $\mathbf{W}^d$ through activations $\mathbf{H}$, which can be formalized as

$$R^d_{n,t} = \sum_k \mathrm{conv}_{\mathbf{H}_{n,k}, \overline{\mathbf{W}}^d_k}(t) \ , \tag{3}$$

where $\mathrm{conv}_{X,Y}(t)$ denotes temporal convolution of X with filter Y at time t. Here, we introduced *normalized basis vectors* $\overline{\mathbf{W}}^d_k$, where the normalization is done jointly over all dimensions d. This normalization is necessary during learning to avoid scaling problems as described in [1].

The energy function in eq. 2 formalizes the standard approximation scheme commonly used for Sparse Non-negative Matrix Factorization. The first term is the distance measure and the second term is a penalization of the overall sum of activations, weighed by the sparseness weight λ. Due to lack of space, we refer to a more detailed explanation in our earlier work [10].

This optimization problem can be solved by alternating the update of one of the factors $\mathbf{H}$ or $\mathbf{W}^d$, while holding the other fixed. Due to the non-negativity of the two factors, the update can be formulated as an exponentiated gradient descent with better convergence properties then pure gradient descent (see e.g. [5]). For a detailed description of the optimization procedure, we refer to [10].

After applying NMF to the data, we have a representation of the input in terms of learned basis vectors and activations. We interpret the basis vectors as features and their corresponding activations as temporal occurrences of those features. For illustrations of the resulting representation, we refer to [10] again.

The full procedure as described above is applied in the training phase for learning the features. During application phase, this optimization is applied again, but the features are given and held fixed and only the activations are updated. Thus only steps 1 to 3 have to be iterated and step 4 is left out.

Classification. As mentioned earlier, we compare our Feature Learning technique to the ones presented in Plötz et al. [7]. To compare the performance of the different techniques, Plötz et al. use the K-Nearest-Neighbor algorithm as a simple classifier. To get comparable results, we also adopt this technique. The K-Nearest-Neighbor algorithm represents a simple but effective standard approach that simply stores all feature vectors from a training set and assigns to a new feature vector the label of the majority of its k nearest neighbors in the feature space. We emphasize that we do not aim at presenting the best possible classifier, but merely want to compare our feature extraction technique to the ones presented earlier.

Applying the classifier to the activations for frame n $\mathbf{H}_n$ (which encodes the temporal positions of features within the frame) directly would yield bad results, because instances of the same class generally differ slightly in the temporal positions of features. Thus, to be temporally invariant within a frame, we sum the activations over the temporal dimension of the frame, yielding the summed activations for each feature as a feature vector that is passed to the classifier.

3 Experiments

Data Sets. We evaluate our method in comparison to the results presented Plötz et al. [7]. The authors tested their methods on four publicly available data sets, which will be described briefly in the following.

Pham et al. [6] describe the data set "Ambient Kitchen 1.0" (AK) consisting of food preparation routines with sensor-equipped kitchen utensils. 20 Persons either prepared a sandwich or a salad, using two kinds of knifes, a spoon, and a scoop, the handle of each of which was equipped with a tri-axial accelerometer. In total, the data consist of 4 hours of recording, sampled at 40Hz, where about 50% cover ten typical food preparation activities.

Huynh et al. [3] describe the dataset "Darmstadt Daily Routines" (DA) consisting of 35 activities of daily living (e.g. brushing teeth, setting the table), captured by two tri-axial accelerometers (one wrist-worn, the other in the pocket) in a lab-like environment. After preprocessing, they yield a sampling frequency of 2.5Hz. In [7] only results for the pocket sensor are presented, hence we also use only the pocket sensor.

Zappi et al. [11] describe the dataset "Skoda Mini Checkpoint" (Skoda) consisting of activities of an assembly-line worker, wearing a number of accelerometers on his arms, while performing ten tasks of quality checks for correct assembly of car parts. The data consists of three hours of recording, sampled at 96Hz. As in [7] we only use a single accelerometers at the right arm.

Roggen et al. [8] describe the dataset for the "Opportunity Gesture Challenge" (Opp) consisting of activities of daily living in a kitchen environment, recorded with multiple accelerometers, body-worn or embedded into objects. Multiple subjects have been recorded on different days. As in [7] we only consider the data of the right arm of the subjects. Also as in [7] we only consider 10 low-level activities and one *unknown* activity class. The data is sampled at 64Hz.

In [7] only a small excerpt of the data is used, consisting of recordings of one subject, because the full data set was not published yet at the time of the publication. Since we have the full data set and the subject used by [7] is left unspecified, it is difficult to get a fair comparison. Thus, for the comparison to [7] to be fairer, we evaluate our method on each single person separately and present the minimum accuracy over all subjects in the dataset.

Before applying SISC, we normalized all datasets by PCA and resampled them to 10Hz, which seemed to be sufficient for activity recognition.

Features. In Plötz et al., four feature extraction techniques are presented, namely Statistical, FFT, PCA, RBM. Further a preprocessing technique based on the empirical cumulative distribution function (ECDF) is used to normalize the data. ECDF is combined with PCA and RBM and called PCA+ECDF and RBM+ECDF. We will describe the methods very briefly here. Please refer to [7] for a deeper explanation.

The method *Statistical* refers to the most commonly used feature extraction method, which is to extract statistical measures, like mean, standard deviation, energy, and entropy over the whole frame. For each sensor, Plötz et al. use x, y, z, pitch, and roll and compute the statistics over each channel independently plus the pairwise correlation between x, y, and z, resulting in a 23-D feature vector for each frame.

The method *FFT* is also widely used and consists of computing for each channel independently the Fourier coefficients through the Discrete Fourier Transform (DFT). Usually only a subset of the resulting Fourier coefficients is used. In [7] the first 10 are used.

In the method *PCA*, features are learned by Principal Component Analysis (PCA). The Eigenvectors corresponding to the largest Eigenvalues are kept as features. In [7] the 30 largest Eigenvectors are used.

The method *RBM* refers to a technique based on Deep Belief Networks. Deep Belief Networks are auto-encoders that use a hierarchy of Restricted Boltzman Machines (RBM) for extracting useful features. It has been shown that Deep Belief Networks can uncover features in the data, which, in turn, can be used for classification. In [7] an architecture consisting of four layers with 1024 Units in each hidden layer and 30 units in the output layer is used.

Additionally to the methods described above, we present results using Shift-invariant Sparse Coding (SISC). For SISC a number of parameters have to be chosen. We applied grid search to find a single set of parameters that gave the highest average classification accuracy for all data sets. The final parameters are as follows: the frame size T is 7 seconds, the width of the basis vectors L is 2.5 seconds (note however that the effective width, i.e. the part of the basis vector that is actually used and above zero, can vary), the number of basis vectors K is 20, and the sparseness parameter λ is 0.1.

In the classification stage of our method we used a K-Nearest-Neighbor classifier with K set to 5. Higher values had no significant impact on the results.

We validated our results by class-balanced 10-fold cross-validation. Regarding computational performance, the learning phase takes up to half an hour on a 2.6GHz Quad-Core CPU for the larger data sets. The application phase takes about 30 milliseconds per frame, which is well within real-time boundaries.

Results. We conducted one experiment devoted to the classification accuracy using the respective feature extraction techniques. In Fig. 1 the classification accuracies for the seven techniques are shown.

The results of the first six techniques are taken directly from [7]. The authors state, that these results are comparable with those published earlier for those data sets. The seventh technique *SISC* is our shift-invariant Sparse Coding approach. Interestingly, *SISC* yields significantly better results on three of the four data sets. We reason that this is mainly due to the shift-invariant nature of this coding technique, which learns features independently of their position in a particular frame and can, in turn, detect a feature even if it is shifted slightly in a frame. Because when a pattern is slightly shifted in a frame, the sum over activations, and hence the feature vector, does not change, which is not the case for PCA and RBM.

In summary, one can use SISC to successfully learn features in an unsupervised manner without prior domain knowledge. Further, SISC outperforms PCA and Deep Belief Networks as a Feature Learning technique in some cases. The caveat, however, is that PCA and Deep Belief Networks are faster during application phase, since they can be applied by simple matrix multiplication, while in SISC a few iterative steps have to be computed for each frame. But, for small frames sizes of up to a few seconds the computational time lies well within realtime boundaries.

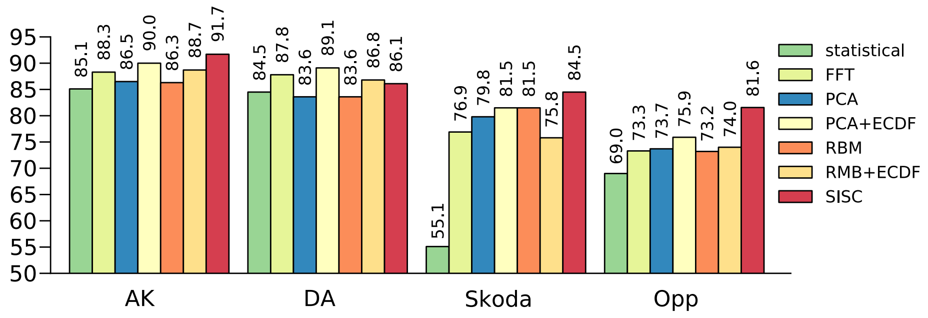


Fig. 1. Classification accuracies of the seven approaches for the four datasets

4 Conclusion

We have presented Shift-invariant Sparse Coding (SISC) as a Feature Learning technique for activity recognition. We compared our method to traditional methods for feature extraction and to two recent approaches for Feature Learning, namely PCA and Deep Belief Networks. The evaluation was performed on four publicly available data sets. The results show that SISC outperforms all other methods on three of the four data sets. Thus, SISC has great potential for application in activity recognition.

Plötz et al. [7] mention that Feature Learning techniques can potentially be used for further sub-frame analysis, which is important if one wants, e.g., to assess certain properties, like the quality of the activities performed. SISC is particularly suited for that task, because the sparse nature of the representation and the shift-invariance allows features to be well localized in time. Thus, the exact position of the features or the relative positions of different features in a frame can be used as a cue for sub-frame analysis.

References

1. Eggert, J., Wersing, H., Korner, E.: Transformation-invariant representation and NMF. In: Proc. of the 2004 IEEE Int. Joint Conf. on Neural Networks, vol. 4, pp. 2535–2539. IEEE (2004)
2. Huynh, T., Schiele, B.: Analyzing features for activity recognition. In: Proc. of the 2005 Joint Conf. on Smart Objects and Ambient Intelligence (2005)
3. Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: Proc. of the 10th Int. Conf. on Ubiquitous Computing, pp. 10–19. ACM Press, New York (2008)
4. Kim, T., Shakhnarovich, G., Urtasun, R.: Sparse Coding for Learning Interpretable Spatio-Temporal Primitives. In: Proc. Neural Inf. Process. Syst (NIPS), vol. 22 (December 2010)
5. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)
6. Pham, C., Olivier, P.: Slice&Dice: Recognizing food preparation activities using embedded accelerometers. In: Tscheligi, M., de Ruyter, B., Markopoulus, P., Wichert, R., Mirlacher, T., Meschterjakov, A., Reitberger, W. (eds.) AmI 2009. LNCS, vol. 5859, pp. 34–43. Springer, Heidelberg (2009)
7. Plötz, T., Hammerla, N.Y., Olivier, P.: Feature Learning for Activity Recognition in Ubiquitous Computing. In: Proc. of the Twenty-Second Int. Joint Conf. on Artificial Intelligence, pp. 1729–1734 (2011)
8. Roggen, D., et al.: Collecting complex activity datasets in highly rich networked sensor environments. In: 2010 Seventh Int. Conf. on Networked Sensing Systems (INSS), pp. 233–240 (June 2010)
9. Smith, E., Lewicki, M.S.: Efficient coding of time-relative structure using spikes. Neural Computation 17(1), 19–45 (2005)
10. Vollmer, C., Eggert, J.P., Groß, H.-M.: Generating Motion Trajectories by Sparse Activation of Learned Motion Primitives. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 637–644. Springer, Heidelberg (2012)
11. Zappi, P., Lombriser, C., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., Tröster, G.: Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 17–33. Springer, Heidelberg (2008)

Hearing Aid Classification Based on Audiology Data

Christo Panchev[1], Muhammad Naveed Anwar[2], and Michael Oakes[1]

[1] Department of Computing, Engineering and Technology, University of Sunderland
St. Peters Campus, Sunderland SR6 0RD, United Kingdom
{christo.panchev,michael.oakes}@sunderland.ac.uk
[2] Knowledge Media Institute, The Open University
Walton Hall, Milton Keynes MK7 6AA, United Kingdom
naveed.anwar@open.ac.uk

Abstract. Presented is a comparative study of two machine learning models (MLP Neural Network and Bayesian Network) as part of a decision support system for prescribing ITE (in the ear) and BTE (behind the ear) aids for people with hearing difficulties. The models are developed/trained and evaluated on a large set of patient records from major NHS audiology centre in England. The two main questions which the models aim to address are: 1) What type of hearing aid (ITE/BTE) should be prescribed to the patient? and 2) Which factors influence the choice of ITE as opposed to BTE hearing aids? The models developed here were evaluated against actual prescriptions given by the doctors and showed relatively high classification rates with the MLP network achieving slightly better results.

Keywords: Audiology Data Mining, Decision Support System, Multi-layer Perceptron, Bayesian Network.

1 Introduction

There is a tremendous growth in the amount of data produced in the medical domain [1] and many approaches, including statistical and neural approaches have been proposed for medical data mining which produce information that helps in problem solving and taking decisions [2,3,4].

The work presented here is based on the large data set of patient records from a major British National Health Service (NHS) audiology centre in England containing 180,000 individual audiology records (from 23,000 patients). The decisions of whether to prescribe an ITE or BTE hearing aid are typically made by audiology technicians working in the out-patient clinics, on the basis of audiogram results and in consultation with the patients. ITE hearing aids are not generally available on the NHS in England, as they are more expensive than BTE hearing aids. However, both types of aids are prescribed at the audiology centre providing the data. Usually the choice is straightforward, but in some cases the technicians could benefit from a second opinion (e.g one given by a decision support system) with an explanation/justification of how that second opinion was arrived at.

© Springer-Verlag Berlin Heidelberg 2013

2 Data Pre-processing

The following attributes were extracted from the raw data:

- Audiograms: the lowest decibel which the patient can hear across a number of frequencies: Air conduction (AC) for 250, 500, 1000, 2000, 4000 and 8000 Hz; and Bone conduction (BC) for 250, 500, 1000, 2000 and 4000 Hz.
- Personal/diagnostic data: Age, Gender, Diagnosis, Tinnitus Masker, Mould, Hearing aid.
- A set of keywords from the doctor's free text notes.

The models presented here were developed and evaluated on the records which had all fields filled for the right ear: AC (air conduction) and BC (bone conduction) thresholds, gender, age and text keywords (5,736 records for training/validation and 1433 records for test), of which 128 also had non-null entries for diagnosis, 98 had non-null entries for masker, and 3983 had non-null entries for mould. In the test set 782 records were given ITE aids, so simply assigning all the patients this type of aid provides 54.6% agreement - referred to as the ZeroR baseline. Since, the data contains only audiograms of patients with ITE/BTE hearing, we do not consider 'no aid class'.

3 Bayesian Network for ITE/BTE Aids

Figure 1 represents the directed acyclic graph for the ITE/BTE aid the Bayesian network obtained from Weka v3.4, where nodes represent the.

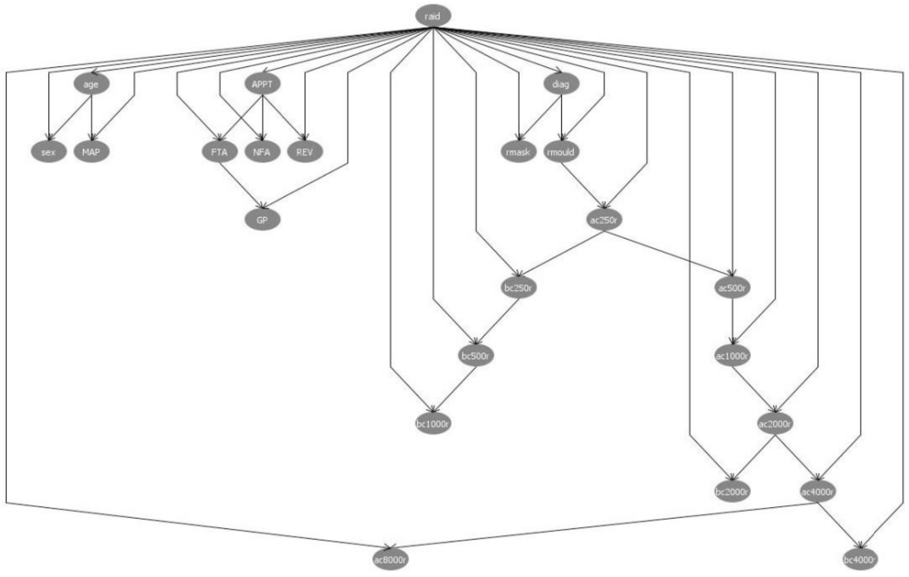


Fig. 1. Directed acyclic graph for ITE/BTE aid

The learning of this network involves finding of edges, that is searching through the possible sets of edges and for each set estimating the conditional probability tables from the data.

The probability tables for ITE/BTE aid, Age and Gender obtained from the nodes (in Figure 1) are given in Table 1 and Table 2. The probabilities for ITE/BTE aid are calculated as 2663/ 5736 = 0.464 for BTE and 3073/ 5736 = 0.536 for ITE, where 2663 and 3073 are the number of instances of BTE and ITE respectively, and 5736 are the total number of instances of ITE/BTE aids.

Table 1. ITE/BTE aid probabilities

ITE	BTE
0.536	0.464

In Table 2, the probability for gender=male, age<=60, and BTE aid, that is, *P(gender = 'male'/age ='<= 60', aid ='BTE')* is calculated as (339 + 1)/(673 + 2) = 0.504, where 339 is the number of instances of “gender=male, age<=60, and BTE aid”, 1 is the initial count for “gender=male, age<=60, and BTE aid”, 673 is the total number of instances with “age<=60 and BTE aid”, and 2 is the count of different values of gender (that is, male and female). Using the same method the probabilities for the rest of the variables are calculated.

Table 2. Gender probabilities

ITE/BTE aid	Age	Gender	
		Female	Male
BTE	<=60	0.496	0.504
BTE	<=70	0.43	0.57
BTE	<=78	0.555	0.445
BTE	>78	0.71	0.29
ITE	<=60	0.503	0.497
ITE	<=70	0.403	0.597
ITE	<=78	0.494	0.506
ITE	>78	0.669	0.331

Testing of these Bayesian network showed that overall there was 93.2% agreement between the predictions of this model and the actual hearing aid chosen by the audiologist (as given in the “type” field) as shown in Table 3. The agreement rate was higher for patients fitted with ITE aids (97.1%) than for those fitted with BTE aids (88.5%).

Table 3. Confusion matrix of results of Bayesian network for ITE/BTE aids

	Human (expert) decision		
Bayesian network	ITE	BTE	Total
ITE	759 (97.1%)	75 (11.5%)	834
BTE	23 (2.9%)	576 (88.5%)	599
Total	782	651	1433 (93.2%)

Considering the ZeroR baseline of the data, which is 54.6%, the agreements found for ITE and BTE provides a significant boost. The Bayesian network also includes interaction of variables, for example, the variable gender was associated with ITE/BTE aid (Figure 1) and also with age and the associated probabilities for gender are calculated in Table 2. Similarly, other variables (such as, diagnosis, masker, mould, AC250, BC250, etc.) were also found associated as shown in Figure 1.

4 Neural Network Model for ITE/BTE Aid

The second model that was deployed is based on a Multi Layer Perceptron. The network had 21 input and 2 output neurons covering the data attributes (Table 4). The network had 5 hidden neurons with hyperbolic tangent sigmoid activation function and was trained using Levenberg-Marquardt backpropagation [5].

Table 4. Input and output attributes of the neural network

Attribute	Value attribute	Values
Age		0 - 78
Gender	Male	0, 1
	Female	0, 1
Diagnosis		-1, 0, 1
Mould	2107	0, 1
	V1	0, 1
	2107V1	0, 1
	2112	0, 1
	Other	0, 1
Frequency	AC250	0 - 75
	AC500	0 - 75
	AC1000	0 - 75
	AC2000	0 - 75
	AC4000	0 - 75
	AC8000	0 - 75
	BC250	0 - 75
	BC500	0 - 75
	BC1000	0 - 75
	BC2000	0 - 75
	BC4000	0 - 75
Mask		0, 1

Output:	Values
ITE	0, 1
BTE	0, 1

Table 5 presents the confusion matrix of the results from the neural network. Although the neural network shows slightly higher overall performance (93.7%), the results between the two models are qualitatively the same. As in the Bayesian model, the highest agreement between the neural network and the medical expert is for the ITE aids (98.2%). The highest misclassification of the models is for the case where the model suggests an ITE aid whereas the human decision was to prescribe the BTE one (11.7%). This is a partially expected result since, as mentioned earlier in the

Table 5. Confusion matrix of results of Neural network for ITE/BTE aids

	Human (expert) decision		
Neural network	ITE	BTE	Total
ITE	768 (98.2%)	76 (11.7%)	844
BTE	14 (1.8%)	575 (88.3%)	589
Total	782	651	1433 (93.7%)

paper, ITE aids are generally not available on NHS in England and doctors have the tendency to bias their decisions toward the generally available BTE hearing aids.

5 Attribute Significance for ITE/BTE Classification

Following the results presented above, the importance of each of the input attributes was evaluated for their relative contribution to the correct ITE/BTE decision. The network trained with the full set of input features were evaluated in separate tests where one of the input attributes was set to 0. The relative importance of an attribute was calculated as proportional to the neural network's misclassification error during tests with the data of that factor being ignored. The output error was calculated over the entire dataset (i.e. training, validation and test data). When a particular input factor is removed, a higher output error will indicate that this attribute is more significant in the performance of the model, i.e. higher importance/effect on ITE/BTE classification, whereas a lower error would indicate relatively lesser degree of relevance.

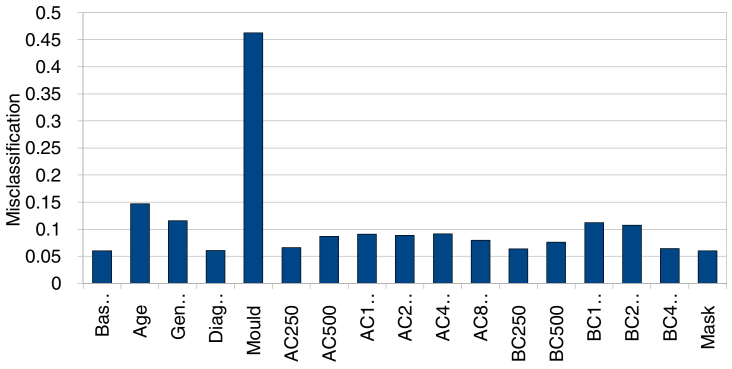


Fig. 2. Relative attribute importance of the input attributes toward the ITE/BTE classification. The base line is 6.0% misclassification error with all input attributes present.

The results presented in Figure 2 show that the Mould is the most significant factor in determining the ITE/BTE aid. This is another expected result, since medically the mould type is highly correlated to the hearing aid being used. Leaving the mould aside, the other significant attributes which can be identified are the Age and Gender causing increased misclassification errors of 12.6% and 11.5% respectively. On the frequencies range, the most significant attributes are shown to be the bone conduction frequencies BC1000 and BC2000 increasing the error to 11.2% and 10.7% respectively. These results are similar to [6] where using logistic regression Age was not found significant factor associated with ITE/BTE hearing aids but Gender, BC1000 and BC2000 were found significant.

6 Conclusions

We have presented two machine learning models for the classification of ITE/BTE hearing aids based on audiology data. Both models provide qualitatively similar results with the Neural Network having slightly better classification rate, indicating that they are both viable for implementation into a real decision support system for hearing aid prescriptions. In addition, the disagreement rate between the models and the audiology experts could provide a quantifiable measure as to what percentage of patients could have benefited from prescribing the appropriate hearing aid based purely on diagnostic data rather than considering the availability and costs of the devices. Furthermore, the discovery of significant attributes (factors) and relationships in audiology data for hearing aid classification will provide supplementary information for audiology experts.

References

1. Ananiadou, S.: Text mining for biomedicine. In: Prince, V., Roche, M. (eds.) Information Retrieval in Biomedicine, Natural Language Processing for Knowledge Integration, Medical Information Science Reference, pp. 1–9. IGI Global, Prince (2009)
2. Bakar, A.A., Othman, Z., Ismail, R., Zakari, Z.: Using the rough set theory for mining the level of hearing loss diagnosis knowledge. In: International Conference on Electrical Engineering and Informatics, Selangor, Malaysia, pp. 7–11 (August 2009)
3. Shalvi, D., DeClaris, N.: An unsupervised neural network approach to medical data mining techniques. In: Proceedings of IEEE World Congress on Computational Inetelligence Neural Networks, pp. 171–176 (1998)
4. Thompson, P., Zhang, X., Jiang, W., Ras, Z.W.: From mining tinnitus database to tinnitus decision support system, initial study. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 203–206 (2007)
5. Hagan, M.T., Menhaj, M.: Training feed-forward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks 5(6), 989–993 (1999)
6. Anwar, M.N., Oakes, M.P.: Data Mining of Audiology Patient Records: Factors Influencing the Choice of Hearing Aid Type. Journal of BMC Medical Informatics & Decision Making 12(suppl. 1), S6 (2012)

BLSTM-RNN Based 3D Gesture Classification

Grégoire Lefebvre[1], Samuel Berlemont[1,2], Franck Mamalet[1], and Christophe Garcia[2]

[1] Orange Labs, R&D, France
{firstname.surname}@orange.com
[2] LIRIS, UMR 5205 CNRS, INSA-Lyon, F-69621, France
{firstname.surname}@liris.cnrs.fr

Abstract. This paper presents a new robust method for inertial MEM (MicroElectroMechanical systems) 3D gesture recognition. The linear acceleration and the angular velocity, respectively provided by the accelerometer and the gyrometer, are sampled in time resulting in 6D values at each time step which are used as inputs for the gesture recognition system. We propose to build a system based on Bidirectional Long Short-Term Memory Recurrent Neural Networks (BLSTM-RNN) for gesture classification from raw MEM data. We also compare this system to a geometric approach using DTW (Dynamic Time Warping) and a statistical method based on HMM (Hidden Markov Model) from filtered and denoised MEM data. Experimental results on 22 individuals producing 14 gestures in the air show that the proposed approach outperforms classical classification methods with a classification mean rate of 95.57% and a standard deviation of 0.50 for 616 test gestures. Furthermore, these experiments underline that combining accelerometer and gyrometer information gives better results that using a single inertial description.

Keywords: LSTM-RNN, DTW, HMM, MEM, hand gesture recognition.

1 Introduction

Accelerometers and gyrometers are nowadays present in our everyday Smartphones. These sensors capture hand movements when users grasp their devices. We can consider two main issues: posture recognition and symbolic gesture recognition. In the first case, the user maintains a posture during a certain period of time, describing for instance the fact that the device is upside down. In the second situation, the user may produce a gesture to execute a system command, like drawing a *heart* symbol in 3D space to call its favorite phone number. Dynamic gesture recognition based on inertial sensors is a very challenging task. Algorithms are confronted to numerous factors causing errors in the recognition process: dynamical differences (intensive versus phlegmatic gestures), temporal differences (slow versus fast movements), physical constraints (device weight, human body elasticity, left or right-handed, seated or standing up, on the move, etc.), classification constraints (mono versus multi users, open or closed world paradigm, etc.). Classically, several steps operate from signal data preprocessing

© Springer-Verlag Berlin Heidelberg 2013

to gesture classification with some intermediate steps like data clustering and gesture model learning. The preprocessing steps aim at reducing the input signals that characterize the corresponding gestures. Different methods can then be applied: calibration, filtering, normalization or vectorization. Data clustering is often applied to reduce the input space dimension and find class referent gesture vectors. A learning phase of a gesture model follows this clustering step and finally a decision rule or a specific classifier is built to label the input data as a recognized gesture or an unknown gesture. In this article, we propose to learn an efficient gesture classifier without any preprocessing method (i.e. from raw MEM data) using a BLSTM-RNN model.

This paper is organized as follows. In Section 2, sensor-based gesture recognition is described with a survey. Section 3 presents our recognition method. Section 4 describes the experimental results. Finally, conclusions are drawn.

2 Accelerometer Based 3D Gesture Recognition

3D gesture recognition using accelerometers has been studied in recent years, and for gesture classification three main strategies stand out which are based on statistics, on geometry or on boosting classifier approaches.

The first strategy has been deeply studied in the last decade with two main approaches: discrete versus continuous HMM [6–8, 11]. Hofmann et al. [6] proposed to use discrete HMM (dHMM) for recognizing dynamic gestures thanks to their velocity profile. This approach consists of two levels and stages of recognition: a low-level stage essentially dividing the input data space into different regions and assigning each of them (i.e. creation of a vector codebook), and a high-level stage taking the sequences of vector indexes from the first stage and classifying them with discrete HMM. The experiments are built using a training set with 10 samples per gesture, each sample representing hand orientation, acceleration data and finger joint angle. A vector codebook is obtained by an input space clustering method (i.e. K-means algorithm). Clustering essentially serves as an unsupervised learning procedure to model the shape of the feature vector distribution in the input data space. Here, the number of HMM states vary from 1 to 10 and the observation alphabet size equals to 120. The comparison between ergodic HMM and left-to-right HMM shows similar results with 95.6% correct recognition rate for 100 gestures. Similar results are presented in [7, 8]. Kallio et al. [7] use 5 HMM states and a codebook size of 8 for 16 gestures. The authors highlight that the performances decrease when using 4 sequences for training the system compared to 20 sequences. The recognition rate falls from 95% to 75% even for this mono-user case study. In [8], a 37 multi-user case is studied with 8 gestures, evaluating the effect of vector quantization and sampling. A rate of 96.1% of correct classification is obtained with 5 HMM states and a codebook size of 8. However, this study can be seen as biased since the K-means clustering is performed from all the available data set and not only the training database. In opposition to the previous studies, and to take into consideration that gesture data are correlated in time, Pylvänäinen proposes in [11] to build a system based on continuous HMM (cHMM). Again, the results are convincing, with 96.76% on a dataset providing 20 samples for 10 gestures realized by 7 persons.

The second strategy for recognizing 3D gestures is based on geometric models with distance computation. The goal is to provide a gallery of some gesture references to model each gesture class and design a decision rule for a test gesture regarding the respective distance to these referent instances. On the contrary to the HMM strategy, no learning phase is needed but computational time is required for a test gesture to be compared to all referent instances. Consequently, the main drawback of this approach is the necessity to find the most relevant samples to represent a gesture class while keeping the number of these referents low in order to minimize the final evaluation processing time. Wilson et al. in [13] compare Linear Time Warping (LTW) and Dynamic Time Warping (DTW) to the HMM based strategy. Their experiment with 7 types of gesture from 6 users shows an advantage for HMM with 90% in opposition to the score of LTW and DTW of respectively 40% and 71%. Liu et al. experiment with more success the DTW strategy in [9]. Gesture recognition and user identification are performed with good recognition rates of respectively 93.5% and 88%. The authors introduce an averaging window of 50 ms for reducing noise and erratic moves. The gesture data, performed over multiple days, consists of 30 samples of 8 gestures for 8 individuals and the user recognition results are obtained from 25 participants. Likewise, in [2], Akl et al. use DTW and affinity propagation for dimension reduction for recognizing 3D gestures. 7 subjects participated producing 3700 gesture traces for a good classification rate of 90%.

The third strategy for recognizing 3D gestures is to learn a specific classifier. Hoffman et al. (see [5]) improve 3D gesture recognition with a linear classifier and Adaboost, inspired by the method proposed in [1] for 2D symbol writer recognition. The experiments show an accuracy of 98% for 13 gestures made by 17 participants. Other studies focus on SVM (i.e. Support Vector Machine) like in [14]. This study uses frame-based descriptors. Each gesture is divided into segments where are computed to form descriptors: mean, energy, entropy, standard deviation and correlation. These descriptors constitute the feature vector to be classified by a multi-class SVM. The obtained results are 95.21% of good recognition for 12 gestures made by 10 individuals.

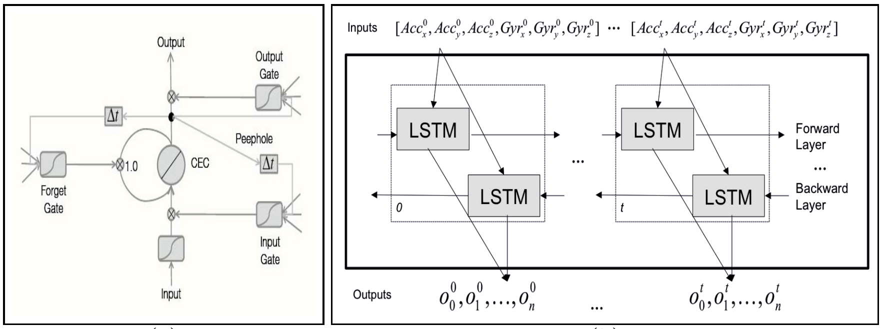
Consequently, many strategies are explored with different paradigms and specific data processing methods on different databases. Nevertheless, theses approaches suffer from finding automatically the relevant parameters (e.g. signal processing, etc.) to deal with gesture variabilities. We develop hereafter our 3D gesture recognition method based on BLSTM-RNN from raw input data and compare it with classical methods on a common database.

3 The Proposed 3D Gesture Recognition Method

3.1 Bidirectional Long Short-Term Memory RNNs

Classical RNNs are a common learning technique for temporal analysis of data since they are able to take into consideration the temporal context. This is achieved by using recurrent connections within the hidden layer which allow the network to *remember* a state representing the previous input values. However, Hochreiter and Schmidhuber in [12] have shown that if RNNs can handle short-time lags between inputs, the problem of *exponential error decay* prevent them from tackling real-life long-term dependencies. They introduced thus the Long Short Term Memory RNNs, that allows a constant error signal propagation through time using a special node called *constant error carousel* (CEC) and multiplicative gates (Fig 1.a). These gates are neurons that can set (input gate), reset (forget gate) or hide (output gate) the internal value of the CEC according to neuron input values and context.

LSTM-RNNs have proven their great ability to deal with temporal data in many applications (e.g. phoneme classification [4], action classification [3]). In this paper we consider gesture data using 6D input vectors through sampling timestep. These data are correlated during the user gestural production, and time lags between the beginning and the end of gesture can be long. For these reasons, LSTM-RNN is chosen to classify the input MEM data sequence. Furthermore, since gesture recognition, at a given timestep, may depend on past and future context, we use Bidirectional LSTM-RNN (BLSTM-RNN), introduced in [4], that consists in two separate hidden layers, the forward (resp. backward) layer able to deal with past (resp. future) context. The output layer is connected to both hidden layers in order to fuse past and future contexts.


Output
Output Gate
Δt
Peephole
Forget Gate
1.0
CEC
Δt
Input Gate
Input
Inputs
$[Acc_x^0, Acc_y^0, Acc_z^0, Gyr_x^0, Gyr_y^0, Gyr_z^0] \cdots [Acc_x^t, Acc_y^t, Acc_z^t, Gyr_x^t, Gyr_y^t, Gyr_z^t]$
LSTM
LSTM
0
...
LSTM
LSTM
t
Forward Layer
...
Backward Layer
Outputs
$o_0^0, o_1^0, \ldots, o_n^0$
...
$o_0^t, o_1^t, \ldots, o_n^t$

(a) (b)

Fig. 1. (a) LSTM neuron. (b) BLSTM-RNN Architecture.

3.2 BLSTM-RNN Architecture, Training and Decision Rule

The proposed gesture classification scheme based on BLSTM-RNN is described in Figure 1.b. First, the input layer consists in the concatenation of accelerometer and gyrometer information synchronized in time (i.e. 6 input values per timestep). Notice that our system relies only on the raw MEMs data, without any preprocessing in opposition to most of state-of-the-art methods. These data are linearly normalized between -1 and +1 according to the maximum value that sensors can provide. The forward and backward LSTM hidden layers are fully connected to the input layer and consist in 100 LSTM neurons each with full recurrent connections. The output layer has a size equals to the number of gesture to classify. The SoftMax activation function is used for this layer to give network responses between 0 and 1 at every timestep. Classically, these outputs can be considered as posterior probabilities of the input sequence to belong to a specific category at a given timestep. This network is learned using classical on-line backpropagation through time with momentum (i.e. learning rate $5e{-}4$, momentum 0.2), as described in [12], on a training set, by targeting the same corresponding gesture class at each time step for each input example. For evaluation of a new gesture sequence, we use a majority voting rule over the outputs along the sequence (i.e. keeping only the most *probable* class at each time step) to determine the final gesture class.

4 Experimental Results

There is no public dataset for comparison of 3D gesture recognition. Therefore, we have collected our 3D gesture dataset to compare classification methods. Our dataset has been captured on an Android Nexus S Samsung device. 22 participants, from 20 to 55 years old, all right-handed, performed 5 times each of the 14 symbolic gestures. This corresponds to 1540 temporal segmented gestures. The sampling time for accelerometer and gyroscope capture is 40 ms. The 14 symbolic gestures are divided into 2 families: linear gestures (e.g. *north*, *south*, *east* and *west flicks*, and *up*, *down*, *pick* and *throw* gestures) and curvilinear gestures (e.g. *alpha*, *heart*, *letter N*, *letter Z*, *clockwise* and *counter-clockwise*). These choices make the dataset difficult. There are classically confusions between *flick* gestures and *letter N* and *Z*. Likewise, the *clockwise* movement is often confused with *alpha* or *heart* symbols. Hereafter, we use temporal segmented gestures where only useful data are efficient to classify the inputs.

We use 3 different configurations to compare our solution based on BLSTM-RNN to 3 state-of-the-art solutions: DTW, dHMM and cHMM based methods. The DTW solution uses a 5 nearest neighbor classification [10] and the HMM solution uses the maximum of likelihood as a decision rule . In all experiments, we use a filtered and vectorized gestural information for these methods and raw MEM information for LSTM solution. In the following, we use a 3-fold cross validation.

The first configuration (DB1) corresponds to the personalization paradigm, where only one user is considered with few learning examples. For this configuration we have used the 70 gestures of a single participant in the learning phase, and ask him to process 16 more instances of each gesture for test (i.e. 224 gestures). The second configuration (DB2) uses 3 instances of each gesture per user for the learning phase: 924 gestures (i.e. 60% of all data) are used for the learning phase and 616 gestures (i.e. 40%) for the test phase. This case corresponds to a multi-user system and a closed world paradigm. The third configuration

(DB3) is composed of all samples from 17 users (i.e. 1190 gestures) and the test data uses the other available gestures (i.e. 350 gestures from unknown users). This case is close to a real system trained with a few examples and having to generalize to new users who want to use it without any personalization phase. Here, the configuration represents the open world paradigm.

Table 1. Good classification rates on DB1, DB2 and DB3

Databases	DB1	DB2	DB3
Methods	Mean & Standard Deviation		
DTW acc	99.40% ± 0.21%	92.59% ± 0.20%	90.29% ± 2.07%
DTW gyro	95.39% ± 0.56%	80.63% ± 2.39%	79.81% ± 1.72%
DTW acc+gyro	99.70% ± 0.42%	94.04% ± 0.15%	91.71% ± 1.46%
dHMM acc	77.14% ± 5.18%	64.09% ± 1.60%	63.81% ± 0.58%
dHMM gyro	57.50% ± 3.24%	43.13% ± 2.35%	49.05% ± 1.15%
dHMM acc+gyro	81.02% ± 3.72%	69.46% ± 2.11%	66.95% ± 1.87%
cHMM acc	99.02% ± 0.81%	83.99% ± 1.09%	80.09% ± 2.82%
cHMM gyro	95.05% ± 2.62%	70.92% ± 0.74%	70.76% ± 0.58%
cHMM acc+gyro	**99.86% ± 0.02%**	85.79% ± 0.67%	82.76% ± 1.41%
BLSTM-RNN acc	84.15% ± 0.67%	94.86% ± 1.23%	89.42% ± 2.45%
BLSTM-RNN gyro	68.90% ± 4.85%	83.39% ± 0.65%	74.19% ± 1.55%
BLSTM-RNN acc+gyro	86.75% ± 0.75%	**95.57% ± 0.50%**	**92.57% ± 2.85%**

Classification Results. Table 1 outlines the global performances of each classifier for configurations DB1, DB2 and DB3 coupling or not accelerometer and gyrometer data. Considering coupled input data (accelerometer+gyroscope), this table shows that our BLSTM-RNN based classifier gives the best results on DB2 and DB3, with respectively $95.57 \pm 0.50\%$ and $92.57 \pm 2.85\%$.

In the three configurations, the dHMM solution provides lower performances which is mainly due to the input data variability and the complexity to determine an automatic discriminant codebook.

On two configurations (DB2 and DB3), the DTW solution achieves the second best performance in mean recognition rate before the cHMM based one.

On DB1 configuration, DTW and cHMM achieve equivalent performances while our BLSTM-RNN approach is less efficient. This is mainly due to the lack of learning data which leads to the classical over-fitting issue. The attempts made with smaller LSTM networks did not allow any improvement on generalization.

When comparing these methods using a single input MEM sensor (accelerometer or gyroscope), we can see that using only gyroscope data is less efficient than using single accelerometer data. Moreover, when these two information are combined, the performances increase with respectively $99.70 \pm 0.42\%, 94.04 \pm 0.15\%$ and $91.71 \pm 1.46\%$, for instance, for the DTW based method on DB1, DB2 and DB3 configurations.

Main conclusions of a deep analysis of confusion matrices (not provided here due to lack of space) are the following. The main drawback for the cHMM based method in this context is the incorrect classification of the N gestures with only

0.95% of correct classification. 62.86% of the N gestures are confused with the pick gestures. A strong confusion appears with opposite gestures as pick and throw or down and up gestures. Opposite gestures may be mis-classified when some user anticipate a north flick gesture by slightly moving back the device in the beginning of the production. On the contrary, the DTW based method provide a good solution to classify linear gestures except for the *throw* gesture which is often recognized as *east* and *north flicks*, which can be explained by the similar nature of production of these three gesture types. Our BLSTM-RNN approach have some issue to distinguish the *east flick* gesture from the *letter Z* and the *up* gesture from the *letter N*, both sharing the same initial movement. This may be due to the uniform learning target chosen (same class at each time step), or the majority voting scheme in recognition phase.

Table 2. Computing time (in ms) to classify one unknown gesture

Databases	DB1	DB2	DB3
Leaning samples	70	924	1190
Test samples	224	616	350
DTW accgyro	11.93 ±0.02	34.57 ±0.47	44.58±0.38
dHMM accgyro	18.31 ±0.17	24.84 ±0.32	16.18±0.32
cHMM accgyro	42.53 ±1.97	23.89±2.74	30.19±1.65
BLSTM-RNN accgyro	30.47± 0.23	31.12±0.57	29.56±0.48

Computing Times. Table 2 presents the computing times for all methods for the 3 configurations in recognition phase executed on an Intel Core i5 CPU at 2.67 GHz with 3.42 Go of RAM. These experimental results show that the computing time for the BLSTM-RNN and HMM based solutions is quite constant regarding the tasks on the different database (i.e. around 30 ms for BLSTM-RNN and 18 ms for dHMM to classify one input gesture for DB1). The learning process is built indeed off-line and consequently the recognition process is fast. On the contrary, the DTW solution requires to compare the input gesture with all learning reference samples. That is why the computing time increases in mean from 11.93 ms for 70 learning samples to 44.58 ms for 1190 learning samples. The DTW solution requires a small number of reference gestures and which makes it hard to cover all user gesture variations. Consequently, the proposed system, based on BLSTM-RNN, achieving the best result performances in multi-user configuration with a recognition computing time independent of training dataset size is a very challenging solution.

5 Conclusion and Perspectives

In this paper, we have presented a contribution based on BLSTM-RNN and a comparison for inertial MEM based gesture recognition. This study about symbolic gesture recognition compares our contribution to 3 classical pattern recognition methods: the geometric approach using DTW and the statistical method based on dHMM and cHMM. We have shown that on multi-user configuration our approach achieves the best mean classification rates, up to 95.57%, in a closed world configuration. Main remaining confusions with the proposed solution are when two 3D trajectories are similar or share some initial movements, as an east flick and a Z letter. New approach using a modified objective function, such as Connectionist Temporal Classification [4], that permits to jointly learn to localize and classify events in input sequences, might be used to overcome this issue or to classify non segmented gestures.

References

1. A practical approach for writer-dependent symbol recognition using a writer-independent symbol recognizer. IEEE Trans. PAMI 29(11), 1917–1926 (2007)
2. Akl, A., Valaee, S.: Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing. In: ICASSP (2010)
3. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Sequential deep learning for human action recognition. In: Salah, A.A., Lepri, B. (eds.) HBU 2011. LNCS, vol. 7065, pp. 29–39. Springer, Heidelberg (2011)
4. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks (18), 5–6 (2005)
5. Hoffman, M., Varcholik, P., LaViola, J.: Breaking the status quo: Improving 3d gesture recognition with spatially convenient input devices. In: Virtual Reality Conference (VR), pp. 59–66 (2010)
6. Hofmann, F.G., Heyer, P., Hommel, G.: Velocity profile based recognition of dynamic gestures with discrete hidden markov models. In: Wachsmuth, I., Fröhlich, M. (eds.) GW 1997. LNCS (LNAI), vol. 1371, pp. 81–95. Springer, Heidelberg (1998)
7. Kallio, S., Kela, J., Mantyjarvi, J.: Online gesture recognition system for mobile interaction. Systems, Man and Cybernetics 3, 2070–2076 (2003)
8. Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Marca, D.: Accelerometer-based gesture control for a design environment. Personal Ubiquitous Comput. 10(5), 285–299 (2006)
9. Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uwave: Accelerometer-based personalized gesture recognition and its applications. In: IEEE PerCom, pp. 1–9 (2009)
10. Petit, E.: GRASP: Moteur de reconnaissance de gestes. Technical report, France Télécom R&D (2007)
11. Pylvänäinen, T.: Accelerometer Based Gesture Recognition Using Continuous HMMs. In: Marques, J.S., Pérez de la Blanca, N., Pina, P. (eds.) IbPRIA 2005. LNCS, vol. 3522, pp. 639–646. Springer, Heidelberg (2005)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation (9), 1735–1780 (1997)
13. Wilson, D.H., Wilson, A.: Gesture recognition using the xwand. Technical Report CMU-RI-TR-04-57, Robotics Institute (April 2004)
14. Wu, J., Pan, G., Zhang, D., Qi, G., Li, S.: Gesture recognition with a 3-D accelerometer. In: Zhang, D., Portmann, M., Tan, A.-H., Indulska, J. (eds.) UIC 2009. LNCS, vol. 5585, pp. 25–38. Springer, Heidelberg (2009)

Feature Selection for Neural Network-Based Interval Forecasting of Electricity Demand Data

Mashud Rana[1], Irena Koprinska[1], and Abbas Khosravi[2]

[1] School of Information Technologies, University of Sydney, Sydney, Australia
{mashud,irena}@it.usyd.edu.au
[2] Centre of Intelligent Systems Research, Deakin University, Geelong, Australia
abbas.khosravi@deakin.edu.au

Abstract. We consider feature selection for interval forecasting of time series data. In particular, we study feature selection for LUBEX, a neural-network based approach for computing prediction intervals and its application for predicting future electricity demands from a time series of previous demands. Our results show that the mutual information and correlation-based feature selection methods are able to select a small set of lag variables that when used with LUBEX construct valid and stable prediction intervals (coverage probability of 97.44% and 96.68%, respectively, for confidence level of 90%). In contrast, the popular partial autocorrelation feature selection method fails to do this (coverage probability of 69.69%). Our evaluation was conducted using one year of half-hourly Australian electricity demand data.

Keywords: Electricity demand forecasting, prediction intervals, neural networks, feature selection.

1 Introduction

Forecasting future electricity demands is an important task for organizations managing and operating electricity networks and electricity markets. In this paper we consider *interval forecasting* for electricity demand data, i.e. at time t the task is to predict an interval of values for time $t+h$ with a certain probability. In contrast, *point forecasting* aims to predict a single demand value for time $t+h$. More formally, our task can be stated as follows: given a time series of n previous half-hourly electricity demands $X_1, X_2, \ldots, X_n$, forecast a Prediction Interval (PI) for the next value of the series X_{n+1}. A PI consists of lower and upper bounds, L and U, between which the future value is expected to lie with a minimum probability $\mu = (1-\alpha)100\%$. Thus, a valid PI for X_{n+1} will satisfy the condition: $P(L(X_{n+1}) \leq X_{n+1} \leq U(X_{n+1})) \geq \mu$.

The great majority of existing approaches for electricity demand forecasting are concerned with point forecasting [1-3]. Although interval forecasting is very useful for risk management in applications requiring balancing of demand and supply [4] such as electricity markets, it still hasn't received enough attention.

A method for predicting PIs using Neural Networks (NN), called LUBE, was recently proposed in [5]. To predict PIs for new data, it uses the point forecasts for the training data and a novel cost function that is minimized during training. LUBE was

© Springer-Verlag Berlin Heidelberg 2013

compared with other NN-based methods for PI construction such as delta [6], Baysian [7] and bootstrap [8] and was shown to generate high quality and valid PIs, typically outperforming the other methods. In [9] we proposed an extension of LUBE, called LUBEX, which utilizes an ensemble of NNs instead of a single NN to reduce sensitivity to NN architecture, weights initialization and perturbation during training.

The focus of this paper is feature selection for interval forecasting of electricity demand data. The few existing methods for interval forecasting haven't studied the effect of feature selection on the quality of PIs. In this paper we investigate the performance of three feature selection methods for time series – the widely used Partial Autocorrelation (PA) and the novel Mutual Information (MI) and Correlation-based Feature Selection (CFS) methods - used in conjunction with the interval forecasting approach LUBEX. Our comprehensive evaluation, using Australian electricity data, showed that MI and CFS were successful while PA was not. Thus, our results confirm the importance of employing suitable feature selection methods in order to construct reliable and informative PIs.

2 Data

We use half-hourly electricity demand data for the state of New South Wales (NSW) in Australia for 2010. The data is publicly available at [10] and contains 8,760 samples. We study each month separately in 12 case studies, one for each month.

The data for each case study is divided into three non-overlapped subsets: training D_{train}, validation D_{valid} and testing D_{test}. The split is 50%-30%-20% respectively. D_{train} is used for feature selection and training of NNs, D_{valid} is used to select the best ensemble of NNs and D_{test} is used for performance evaluation.

3 PI Quality Measures

Following [5] we use three measures to evaluate the quality of PIs: Prediction Interval Coverage Probability (PICP), Prediction Interval Normalized Averaged Width (PINAW) and their combination Coverage Width-Based Criterion (CWC). A high quality PI will have a high coverage probability (greater than the predefined confidence level μ) and small width.

PICP. Given a dataset of N examples, PICP is the probability that the target value X_i of the i-th example will fall between the upper U_i and lower L_i bounds of the prediction interval PI_i, averaged over all i. It is calculated empirically as:

$$PICP = \frac{1}{N}\sum_{i=1}^{N} c_i \,.\, 100\ \%, \text{ where } c_i = \begin{cases} 1, & if\ X_i \in [L_i, U_i] \\ 0, & otherwise \end{cases}$$

PINAW. It measures the average width of the PIs, for all points in the dataset, normalized by the range of the target values R: $PINAW = \frac{1}{N\,R}\sum_{i=1}^{N}(U_i - L_i)$

CWC. It combines PICP and PINAW using the parameters η and γ:

$$CWC = PINAW\left(1 + \gamma e^{-\eta(PICP-\mu)}\right), \text{ where } \gamma = \begin{cases} 0, & if\ PICP \geq \mu \\ 1, & otherwise \end{cases}$$

CWC is underpinned by two main principles: 1) If the coverage probability is above the confidence threshold, CWC should depend only on the PI's width. This is achieved by setting γ to 0; CWC becomes equal to the width PINAW and has a low value; 2) If the coverage probability is below the confidence threshold, i.e. the PIs are not valid, CWC should have a high value, regardless of the width. This is achieved by using a high value for η in the exponential term and by setting γ to 1 to consider this term. Due to the high value of the exponential term, the influence of PINAW is lost and CWC becomes high. Thus, CWC balances the PI's usefulness (narrow width) and correctness (acceptable coverage probability). For more details about CWC, see [5].

4 The LUBEX Method

LUBEX is a recently proposed method [9] for computing PIs using an ensemble of NNs. It is an extension of LUBE [5]. A single LUBE NN is a multilayer perceptron with p input neurons, corresponding to the input variables of each example, two output neurons corresponding to the lower and upper PI bounds for this example and one or more hidden layers. A LUBE NN is trained to minimize CWC using the simulated annealing algorithm which combines hill-climbing and random walk. Note that the backpropagation algorithm cannot be applied as CWC is not differentiable. During training, the two targets U_{i+1} and L_{i+1} of PI_{i+1} are both set to the target point forecast X_{t+1}. The trained NN is then used to predict the PIs for the testing data.

Single LUBE NNs are sensitive to the network architecture, random initialization of weights and random perturbation of weights during training. To reduce this sensitivity, LUBEX uses an ensemble method. It considers NNs with one hidden layer and constructs 30 NN architectures $A_1,..,A_{30}$ by varying the number of hidden neurons from 1 to 30. For each architecture A_i, it builds an ensemble E_i of m NNs ($m=100$ in our experiments). The ensemble members of E_i have the same architecture A_i but are initialized to different random weights. Each of them is trained on the training set. To predict the PI for the new example i, E_i combines the predictions of its members by taking the median of their lower and upper bounds: $PI_i = [median(L_{i1},..L_{im}),\ median(U_{i1},\dots,U_{im})]$. The ensembles are evaluated on the validation set, the best one is selected (the one with smallest CWC) and use to predict the test data. In [9] we showed that the use of ensemble is essential; without it LUBEX's performance varied considerably for multiple runs and the PICP was also not satisfactory in many cases.

5 Feature Selection for Constructing PIs

5.1 CFS

CFS [11] is a state-of-the-art filtering algorithm for feature subset selection. It selects a subset S of k features that maximizes the heuristic: $Merit_S = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}}$, $\overline{r_{cf}}$ is the average feature-class variable correlation and $\overline{r_{ff}}$ is the average feature-feature correlation. This heuristic favors subsets containing features that are good individual predictors of the class variable and are also not correlated with each other.

To conduct feature selection using CFS, we apply a 1-week sliding window to the training data resulting in 336 half-hourly lag variables and then use CFS to select a subset of these 336 variables. The selection is done separately for each case study and the selected features are listed in Table 1, with their total number shown in brackets.

Table 1. Selected lag variables using CFS and MI to predict Pi_{t+1}

Case Study	CFS	MI
1 (Jan)	1,46,120,336 (4)	1-4,47-50,335-336 (10)
2 (Feb)	1-3,48,120,217,288,333-335 (10)	1-3,47-49,289,334-336 (10)
3 (Mar)	1,48,115,336 (4)	1-4,48,144,287-288,335-336 (10)
4 (Apr)	1-2,47,121,287,336 (6)	1-3,48-49,287-289,335-336 (10)
5 (May)	1,48,223,335-336 (5)	1-3,48-49,287-288,334-336 (10)
6 (Jun)	1-2,47,158,225,335 (6)	1-4,47-49,288,335-336 (10)
7 (Jul)	1,48,225,255,288,335 (6)	1-3,48-49,144,288,334-336 (10)
8 (Aug)	1,24,48,288,335 (5)	1-3,47-49,287-288,335-336 (10)
9 (Sep)	1,48,208,220,288,335 (6)	1-3,47-49,288,334-336 (10)
10 (Oct)	1-2,119,144,210,286,335 (7)	1-3,48,286-288,334-336 (10)
11 (Nov)	1-3,20,95,192,216,334-336 (10)	1-5,48,49,288,335-336 (10)
12 (Dec)	1-2,48,120,216,335 (6)	1-4,48,287-288,334-336 (10)

5.2 MI

MI measures the interdependence between two variables, both linear and non-linear. If the two variables are independent, their MI is zero; if the two variables are dependent, their MI is positive and the value reflects the strength of the dependency.

Computing MI for continuous variables is more difficult than for nominal variables as it requires assumption about the data distribution. In this paper we apply a method for MI estimation based on *k*-nearest neighbor distances [12]. This method has minimal bias and was shown to be efficient and reliable.

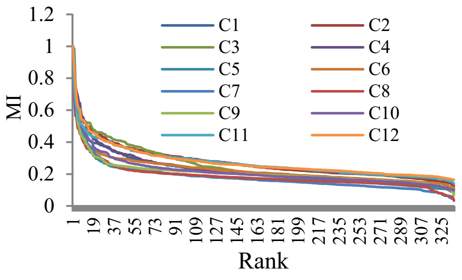


Fig. 1. Feature ranking based on MI

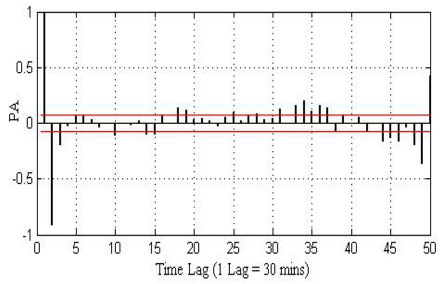


Fig. 2. PACF for case study 1

To conduct the feature selection using MI, we apply again a 1-week sliding window to the training data, compute the MI scores of each of the resulting 336 half-hourly lag variables and then rank these variables in decreasing order based on the MI score. Fig. 1 shows the normalized MI scores for each variable in ranked order. We can see that the graphs for all 12 case studies are very similar - the MI score sharply drops at rank 10 and after that decreases very slowly. Based on these results, we select the 10 highly ranked variables for each case study; they are listed in Table 1.

5.3 PA

PA is a popular feature selection method for time series data. The PA value between two observations X_t and X_{t-h} in a time series is the linear correlation between them, conditional on $X_{t-h+1}, \ldots, X_{t-1}$, the set of observations that come between them. PA only measures linear dependencies; a value close to 1/-1 shows a strong positive/negative dependency while a value close to 0 shows no dependency.

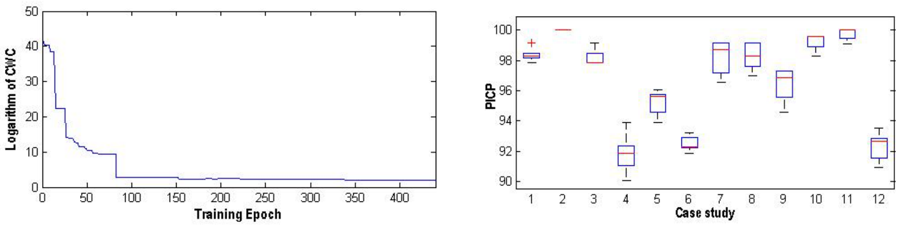
To form a feature set, we compute the PA till lag 336 (1-week sliding window) using the training data, for each case study separately. As an example, Fig. 2 shows the Partial Autocorrelation Function (PACF) for the first 50 lags of case study 1. We then select the lags with PA higher than the confidence threshold (shown with the two parallel horizontal lines in Fig. 2). The selected features are not shown due to space limitation. Their number varied between 39 for case study 3 and 68 for case study 6.

6 Results and Discussion

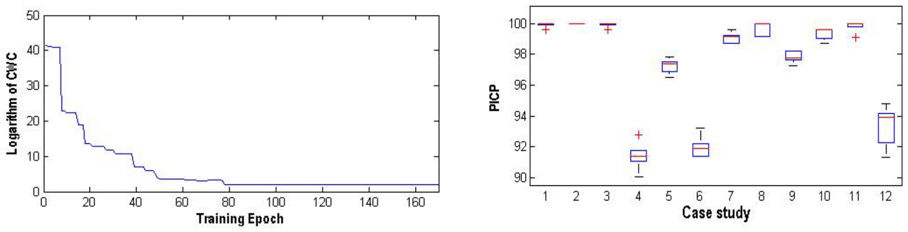
6.1 Convergence

We firstly investigate the LUBEX's convergence on the training data. As an example, the left parts of Fig. 3 show typical convergence of a LUBEX's ensemble member (single NN) using CFS, MI and PA, for case study 2 (February).

We can see that for this example CWC decreases quickly and converges after 100-150 epochs for all feature selectors. However, a closer examination of the data for all NNs and case studies shows that while LUBEX with CFS and MI always converged, LUBEX with PA converged only in about half of the cases.

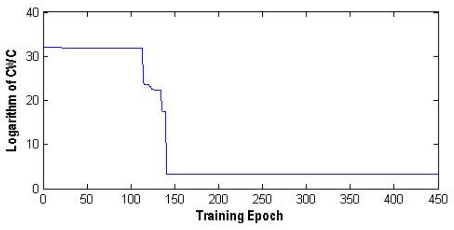


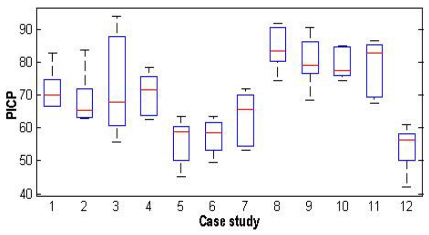
a) LUBEX with CFS



b) LUBEX with MI

Fig. 3. Convergence on training data for case study 2 (left) and variability of PICP (right) for LUBEX with CFS (a), MI (b) and PA (c)





c) LUBEX with PA

Fig. 3. (*continued*)

6.2 Quality of Constructed PIs

To evaluate the quality of the constructed PIs we compute the three performance measures using the testing data: PICP, PINAW and CWC. Table 2 presents the results for LUBEX with CFS, MI and PA.

To further reduce the variability of the NN performance due to the random weight initialization and perturbation during training, for each case study we repeated the experiments five times, i.e. we created and evaluated an NN ensemble five times. The results in Table 2 are the average values over the five runs.

We first examine the coverage probability PICP. We can see that PICP for CFS and MI is higher than the prescribed confidence level (μ=90%) for all case studies, i.e. the constructed PIs are valid. The average PICP over all case studies and the standard deviation are: 96.68±2.76% for CFS and 97.44±3.29% for MI; hence, the constructed PIs considerably outperformed μ. For both CFS and MI, nine out of twelve cases achieve PICP greater than 95% and the remaining three case studies (4, 6 and 12) have PICP between 91.40% and 93.30%. Overall, the PICP is higher when using MI than CFS. In contrast to CFS and MI, the PICP for PA is lower than the prescribed confidence level for all case studies. The average PICP for PA is 69.69±9.45%, which is considerably lower than μ. In summary, LUBEX with CFS and MI as feature selection methods was able to construct PIs with high coverage probability while LUBEX with PA failed to do this.

We now examine the interval width PINAW. We can see than LUBEX generates narrow PIs with all feature selection methods. The average widths are: 6.16±1.29 for PA, 11.86±2.24 for MI and 15.97±3.10 for CFS. Although the PI width for PA is the narrowest, their coverage probability is unacceptable. By comparing CFS and MI we can see that their coverage probabilities for case studies 2, 10 and 11 are very similar but the interval widths for MI are smaller than for CFS. This indicates higher uncertainty for LUBEX with CFS for these case studies.

We finally examine the CWC values. For CFS and MI, these values are the same as the PINAW values. This follows from the definition of the CWC function: when the coverage probability is higher than the prescribed confidence value, CWC=PINAW. By contrast, the CWC values for PA are very high. This is also as expected and follows from the definition of CWC – when the PIs fail to satisfy the required minimum coverage probability, they are invalid regardless of their width; this results in high CWC values as CWC includes a heavy penalty for invalid PIs.

Table 2. PIs for testing data for LUBEX with CFS, MI and PA - performance measures

Case	CFS			MI			PA		
Study	PICP [%]	PINAW	CWC	PICP [%]	PINAW	CWC	PICP [%]	PINAW	CWC
1 (Jan)	98.35	19.27	19.27	99.91	12.55	12.55	71.60	7.31	1.3x10^{16}
2 (Feb)	100.0	19.64	19.64	100.0	17.04	17.04	68.61	9.03	6.4x10^{14}
3 (Mar)	98.17	18.46	18.46	99.91	12.40	12.40	73.16	6.07	1.4x10^{12}
4 (Apr)	91.81	13.76	13.76	91.40	10.33	10.33	70.23	6.54	2.1x10^{16}
5 (May)	95.22	13.19	13.19	97.22	9.69	9.69	55.74	5.08	2.8x10^{14}
6 (Jun)	92.49	11.13	11.13	91.95	8.84	8.84	57.39	4.55	1.5x10^{16}
7 (Jul)	98.18	16.85	16.85	99.04	10.83	10.83	63.03	4.76	1.9x10^{14}
8 (Aug)	98.27	20.55	20.55	99.65	14.06	14.06	84.42	6.82	6.4x10^{15}
9 (Sep)	96.38	18.64	18.64	97.83	12.13	12.13	80.36	7.61	8.1x10^{15}
10 (Oct)	99.22	15.82	15.82	99.30	13.25	13.25	79.57	6.51	5.9x10^{17}
11 (Nov)	99.73	11.65	11.65	99.82	10.23	10.23	78.29	4.96	2.6x10^{11}
12 (Dec)	92.29	12.71	12.71	93.30	11.01	11.01	53.85	4.70	8.3x10^{17}
mean	96.68	15.97	15.97	97.44	11.86	11.86	69.69	6.16	1.2x10^{17}
st.dev.	2.76	3.10	3.10	3.29	2.24	2.24	9.45	1.29	2.6x10^{17}

6.3 Variability of PICP

We also compare the stability of the PICP results for the testing data. The right parts of Fig. 3 show box plots of PICP for LUBEX with CFS, MI and PA, respectively, for the five runs of each case study. Five important values are represented for each case study: the edges of the box show the 25th and 75th percentiles, the whiskers of the box show the minimum and maximum values not considered outliers and the crosses show values that are considered outliers. Note the different scales on the *y* axes.

We can observe that the variation over the five runs is very small for CFS and MI and very high for PA. Comparing CFS and MI, we can also see that CFS is slightly more variable than MI but all variations are above the nominal confidence value. We conclude that LUBEX with CFS and MI generated PIs that were stable over multiple runs while LUBEX with PA generated highly unstable PIs.

7 Conclusion

In this paper we considered the task of constructing NN-based PIs for electricity demand forecasting. We extended the interval forecasting method LUBEX by studying the effect of three feature selection methods on the quality of the generated PIs: the novel CFS and MI and the traditional PA. Our results showed that CFS and MI were able to identify a small set of informative lag variables, that when used with LUBEX resulted in fast convergence during training and high quality PIs for new data. These PIs were valid as they satisfied the minimum coverage probability of 90% (MI: PICP=97.44±3.29%, CFS: PICP=96.68±2.76%) and showed little variation for multiple runs of LUBEX. In contrast, LUBEX with PA did not always converge and produced PIs that were invalid and highly variable (PICP=69.69±9.45%). In addition, CFS and MI selected a considerably smaller set of features in comparison to PA, 4-10 versus 39-68, which means faster training of the NN component and faster prediction for new instances. Overall, the best performance was achieved by LUBEX with MI.

Acknowledgement. Mashud Rana is supported by an Endeavour award and a NICTA scholarship.

References

1. Taylor, J.W.: Triple Seasonal Methods for Short-Term Electricity Demand Forecasting. European Journal of Operational Research 204, 139–152 (2010)
2. Koprinska, I., Rana, M., Agelidis, V.G.: Yearly and Seasonal Models for Electricity Load Forecasting. In: International Joint Conference on Neural Networks (IJCNN), San Jose, pp. 1474–1481. IEEE Press (2011)
3. Chen, Y., Luh, P.B., Guan, C., Zhao, Y., Michel, L.D., Coolbeth, M.A.: Short-Term Load Forecasting: Similar Day-based Wavelet Neural Network. IEEE Transactions on Power Systems 25, 322–330 (2010)
4. Chatfield, C.: Time-Series Forecasting. Chapman &Hall/CRC (2000)
5. Khosravi, A., Nahavandi, S., Creigton, D., Atiya, F.: Lower Upper Bound Estimation Method for Construction of Neural Network-Based Prediction Intervals. IEEE Transactions on Neural Networks 22(3), 337–346 (2011)
6. Hwang, J.T.G., Ding, A.A.: Prediction Intervals for Artificial Neural Networks. Journal of the American Statistical Associatio 92(438), 2377–2387 (1997)
7. MacKay, D.J.C.: The Evidence Framework Applied to Classification Networks. Neural Computation 4(5), 720–736 (1992)
8. Heskes, T.: Practical Confidence and Prediction Intervals. In: Mozer, T.P.M., Jordan, M. (eds.) Neural Information Processing Systems. MIT Press (1997)
9. Rana, M., Koprinska, I., Khosravi, A., Agelidis, V.G.: Prediction Intervals for Electricity Load Forecasting Using Neural Networks. In: International Joint Conference on Neural Networks(IJCNN). IEEE Press, Dallas (2013)
10. AEMO (2013), `http://www.aemo.com.au`
11. Hall, M.A.: Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In: Int. Conference on Machine Learning (ICML), pp. 359–366 (2000)
12. Kraskov, A., Stögbauer, H., Grassberger, P.: Estimating Mutual Information. Physical Review E 69 (2004)

A Combination of Hand-Crafted and Hierarchical High-Level Learnt Feature Extraction for Music Genre Classification

Julien Martel[1], Toru Nakashika[2], Christophe Garcia[1], and Khalid Idrissi[1]

[1] Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR 5205, France
firstname.name@insa-lyon.fr
[2] Department of System Informatics, Kobe University, Japan
nakashika@me.cs.scitec.kobe-u.ac.jp

Abstract. In this paper, we propose a new approach for automatic music genre classification which relies on learning a feature hierarchy with a deep learning architecture over hand-crafted feature extracted from an audio signal. Unlike the state-of-the-art approaches, our scheme uses an unsupervised learning algorithm based on Deep Belief Networks (DBN) learnt on block-wise MFCC (that we treat as 2D images), followed by a supervised learning algorithm for fine-tuning the extracted features. Experiments performed on the GTZAN dataset show that the proposed scheme clearly outperforms the state-of-the-art approaches.

Keywords: music genre classification, high-level hierarchy feature extraction, deep learning, deep belief networks.

1 Introduction

In the last decade, automatic music genre classification has become more important as the digital entertainment industry developed. Now audio files are distributed over the world wide web and are available as digital content with auxiliary-data also called meta-data. In order to search proper music from huge databases, labels in meta-data have to be assigned to each piece beforehand. The point in using a music genre classification system, is to assign them automatically instead of spending lots of effort in manual annotation.

Feature extraction from an acoustic music signal is a significant step in automatic music genre classification. Most systems in the early years mainly relied on timbre features extracted from a windowed short signal, such as MFCC, STFT, LPC, Filterbank Coefficients and Autoregressive Model [1]. Other methods employed statistical models of the timbre features such as histograms, means, variances, etc. [2]. These approaches, however, extract the features frame-by-frame and do not capture any temporal information.

As mentioned in [3], spectral transition in short term is considered to be an important factor for musical genre classification as well as timbre features of the frame. Meanwhile, a block-wise approach, where a feature descriptor involves

© Springer-Verlag Berlin Heidelberg 2013

multiple frames of a few seconds, has been gathering more attention in recent years [4]. One example can be found in [5], where 5 different block-wise features are obtained from 10 or 25 frames and are used for genre classification. Costa *et al.* attempted to extract texture features from a spectrogram of a few seconds, inspired by works in image processing [6]. There, the system extracts 7 statistical texture features after calculating a Gray Level Co-occurrence Matrix (GLCM) [7] from each spectrogram, and classifies the music signals using SVMs.

Another notable block-wise method was proposed by Tom *et al.* [8], where they adopted Convolutional Neural Networks (ConvNets) for bi-dimensional feature extraction and genre classification. In their work, the ConvNets learn music patterns given a bi-dimensional MFCC map and corresponding genre labels in a supervised way.

Besides, other Deep Learning approaches were used as a way to build hierarchical representations. One of the most well-known Deep Learning models is the Deep Belief Network (DBN) [9], which stacks multiple Restricted Boltzmann Machine (RBM) layers hierarchically. Hamel and Eck [10] adopted the DBN to learn high level musical features. In [11], a Sparse Encoder Symmetric Machine (SESM) was proposed by Ranzato *et al.* as another extension of DBN attempting to produce better representations in terms of sparseness. Since such deep learning methods rely on unsupervised feature extraction, it is expected that the models automatically extract more tractable and better-separated features for the supervised classifier.

In this work, we propose and study a deep learning approach for musical genre classification. We fed block-wise "hand-crafted" (by opposition to "learnt") features of MFCCs that are respectively inspired from [6] and [8] into a deep architecture in which we learn a higher level feature hierarchy with an unsupervised learning algorithm and use a supervised learning algorithm for fine-tuning using a set of known labels of songs. Experimental results on the well-known 10 musical genre GTZAN database show the efficiency of our approach.

2 The Proposed Approach

The deep learning method that we propose rely on Deep Belief Networks. Our architecture is based on the following functional blocks:
Hand-crafted features extraction: from the raw audio-file, we extract several block-wise hand-crafted MFCC features;
High-level learnt features extraction: from these features, we extract a hierarchy of high-level learnt features with an unsupervised learning algorithm;
High-level learnt features fine-tuning: we fine-tune these high-level "learnt" features to adapt them to our classification task with a supervised learning algorithm;
Classification: from this fine-tuned high-level hierarchy of features, we train a classifier such that each of the blocks votes for a certain genre;
Voting scheme: we collect the votes in a given voting space (binary, probabilistic scoring etc.) and output a genre associated to the whole music.

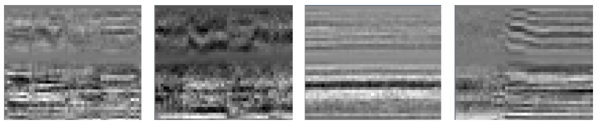


Fig. 1. Examples of block-wise MFCC extracted from the GTZAN database

2.1 Hand-Crafted Feature Extraction with Block-Wise MFCC

The idea behind using hand-crafted features on the "raw" music signal is to process it in a form that will be meaningful for the feature extractor we want to learn in a next step. These hand-crafted features, like Mel-Frequency Cepstral Coefficients (MFCCs) [12], capture a lot of engineering knowledge that has been developed over the years to extract relevant information in audio frames.

We therefore built our block-wise MFCC by computing the cepstrum coefficients over non-overlapping audio frames of 28 milliseconds. The quantized values on the mel-scale (in 40 bins) obtained for a MFCC are concatenated in a block with others computed in the next frame so that it constitutes a time-MFCC domain that we can regard as a bi-dimensional structure (an image). In our experiments, we chose to use 50 blocks so that the block-wise MFCC lasts 1.4 second. By using such block-wise MFCC, we aim at capturing temporal information (in the limit of the block size times the frame length) and timbral information thanks to the MFCC transformation [13]. Some examples of the MFCC blocks we obtain on audio files of the GTZAN database are shown in Figure 1.

2.2 Learnt High-Level Hierarchy Feature Extraction

One of the originalities of this work resides in the way we extract our block-wise features that can both capture timbre and temporal information. These two pieces of information are represented on 2D maps and can be regarded as "images" with a strong bi-dimensional structure. However, we do not directly classify these features with any supervised learning algorithm like in [14] because we know they extract relevant information but which is probably still "drown" and highly hidden in these features. Our idea is to "capture the regularities" in the music genre by first applying a powerful unsupervised learning algorithm which can statistically find a hierarchical structure in the input data. Moreover, one should see the design of these "hand-crafted" features as limited by the complexity human people can possibly put in it and by what they think is interpretable. This is legitimate to try to extract more from these features and combine the resulting objects in a way human would not be able to. Contrary to our method, other recent contributions directly worked on the audio signal by learning Deep Belief Net on raw audio files or on spectrum [10]. Our strategy that learns a hierarchy of features on the top of hand-crafted features performs better than these methods also probably thanks to the use of these hand-crafted features that consists in a first "rough-extraction" in the raw audio data.

In conventional learning methods for neural networks like back-propagation, problems arise when building deep architectures with many layers. If for certain applications a few layers can be sufficient to learn patterns from data, this is clearly not the case in audio especially in a genre classification task where the intra-class variability is very high. The fundamental hope in the proposed deep architecture is to better fit the underlying data with less hidden variables in each layer, more layers allowing to learn intricate "correlation" between these variables. Learning a good set of features, that is able to capture the main regularities of the model without capturing too much noise or "irregularities" make the parametrization of deep learning machines really challenging and subtle. However, they proved in many applications to be able to learn extremely interesting structures in features. In this work, we use an instance of deep learning strategies: the Restricted Boltzmann Machines as a building block for unsupervised learning stacked in Deep Belief Networks we then fine-tune in a supervised fashion.

The Restricted Boltzmann Machine. A Restricted Boltzmann Machine (RBM) is a type of Boltzmann Machine which is an instance of a probabilistic graphical model with interesting properties making inference tractable. The simplified connections result in a bidirectional bipartite graph composed of binary (that can be extended to real valued) stochastic units. On one side, a set of visible units $v = \{v_i\}_V$ receive the sensory input data (our block-wise MFFC), while, on the other side, hidden units $h = \{h_j\}_H$ can be regarded as holding an internal model representation. They are connected together by a set of weights $W = \{w_{ij}\}_{V \times H}$ under the assumption that the weights can be used symmetrically $w_{ij} = w_{ji}\ \forall (i,j) \in V \times H$. Pairwise energies can be defined over the so-formed network and result as an energy for the configuration of the different units: $E(\mathbf{v}, \mathbf{h}) = \mathbf{v}^T W \mathbf{h}$. This energy codes the correlation for any combination of two given binary units (in this case) to be "on" or "off" together. Then, a probability to be in a certain energy state (joint probability of the visible units) can be derived using the Boltzmann-Gibbs Distribution and as an analogy to statistical mechanics: $p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-\frac{1}{T} E(\mathbf{v}, \mathbf{h})}$ where Z stands for the partition function and T is an arbitrary temperature.

For RBMs, it is easy to analytically compute $p(h|v)$ and $p(v|h)$.

Two phases can be distinguished in the learning process: 1) a positive phase: when the visible data is clamped to the visible units and produces the activation of hidden units 2) a negative phase: with free visible units for which the hidden units produce a reconstructed visible data (an hallucination of the model). One possible learning procedure uses unsupervised learning and tries to minimize the reconstruction error between its own input and the reconstruction fantasized by the model. We use the so-called contrastive divergence: a simili negative-log likelihood learning procedure, for which the weights update is $\Delta w_{ij} = \epsilon (< v_i h_j >^+ - < v_i h_j >^-)$ where ϵ is the chosen learning rate and $< \cdot >$ is the expectation — the frequency of having v_i and h_j "on" together — computed respectively during the positive $(+)$ and negative $(-)$ phase.

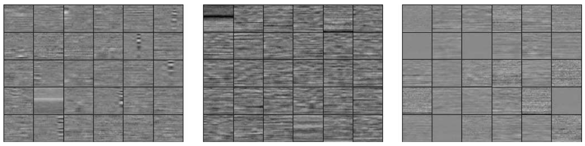


Fig. 2. From right to left : randomly selected features in each layer 1-3 of the hierarchy learnt on block-wise MFCC

Deep Architecture Used in the Experiment Set-up. We build a DBN with three RBM layers stacked on the top of each other and a single layer Perceptron. We clamp the block-wise MFCC on the 2000 units input layer and use 10 Perceptrons as outputs for classification. Concerning the hidden layer architecture we achieve dimensionality reduction by using two layers of 600 units in a row, with a high sparse penalty so that it helps to converge towards interpretable features with condensed information. The last hidden layer contains 2000 units to help the Perceptrons with linear separability. We used Gaussian visible units in the first RBM layer because we intend to use the real values provided by our block-wise MFCC. Gaussian units have shown to be good at learning real valued data and seem much better at properly modeling our problem than binary ones. A corollary is that it then takes much more time to learn such units. Using Gaussian units makes the learning signal theoretically unbounded; therefore the whole learning procedure might diverge very fast. That is why we use a low learning rate of 0.01 with a L_2 regularizer through weight decay to prevent an explosion in the weight values.

In Figure 2, we present some randomly selected features from the three layers of the Deep Belief Network learnt on our block-wise MFCC. To be visualized the feature intensities are shifted and scaled by a normalization process (centered on zero and scaled by their dynamic). Features in the second and third layers are back-projected in the first layer in order to be displayed.

2.3 High Level Learnt Feature Fine-Tuning and Voting Scheme

The fine-tuning step consists in using a supervised learning algorithm with the labels to help the high-level features we extracted previously to converge toward features specific to our classifier. The unsupervised learning procedure we run for RBM, namely "Contrastive-Divergence 1", does not make use of any label. This is a strength because it can be run over big-data for which labels have not been set. In this step, there is no need that the whole data we used previously to be labeled, as we might want to use only a fraction of it to fine-tune the classifier. As we target a 10-genre classification, we use a single layer of 10 Perceptrons on top of our high level feature hierarchy, each of them being set with a hyperbolic tangent activation function which outputs a confidence between -1 (not confident at all) and 1 (almost sure) for the associated audio genre.

2.4 Voting Scheme

The final step in our music-genre classification pipeline is to collect the confidence outputs of the Perceptrons for a single song given the different classification scores for each of the high-level features we extracted from each of the block-wise MFCC. We propose two very simple voting schemes.

One-shot Voting Strategy: In the first case, for each frame from which we extract our features, we perform a max operation over the classifiers (Perceptrons). The classifier with the most confident output for a certain genre casts a (+1) in a voting space consisting of the various genres, the other ones do not impact the vote. The genre which received the highest number of votes is declared the "winner" for the song.

Scoring Strategy: In the second scheme, each classifier votes according to its confidence in a genre. The confidence is scored by the output of the Perceptrons. Analogously the genre receiving the highest confidence is chosen to be the winner for the analyzed song. In our system, as the final Perceptrons are set with hyperbolic tangent activation functions, their scores range between -1 and 1 for each genre.

3 Experiments and Results

We conducted 10-musical-genre-classification experiments using the GTZAN dataset [1], which is widely used in this task. The dataset contains 100 songs for each of the following musical genres: Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae and Rock (1000 songs in total). Some of them, —and we will experience it— are rather easily identifiable: it is usually the case of Reggae, Classical and Hard-rock. They usually do not give a lot of false negatives and have therefore a high recall rate. However, other genres are difficult to classify. For instance, the inter-class variability between rock and hard-rock is likely to be small against their intra-class variability and many false-positives of rock are recognized as hard-rock leading to a low precision. In GTZAN, each song is recorded during 30 seconds with a sampling rate of 22050 Hz using 16 bits quantization. We use a 10-fold cross validation for evaluating our method. For a fold, we select 90 songs from each genre for the training set (in total 900 songs) and the rest is used for the validation (100 songs). In order that all the songs are once used for training and validation, we proceed in such a manner 10 times and we average our results over the 10 trials. In table 1, we present the results of the classification after the fine-tuning step and after the voting scheme. The results clearly show that applying such a classification scheme provides very high classification rates: 96.0% of average recall rate on block-wise MFCC (before voting) and 99.8% on entire songs after voting with the score voting strategy. Table 2 clearly shows the superiority of the proposed approach when compared to the state-of-the-art approaches, with an important gap of 7.1% with the second best method.

Table 1. Per class statistics: Recall, Precision and F_1-measure on the block-wise MFCC after high-level features fine-tuning and Recall for the final voting scheme (2 strategies) using the fused high-level features from block-wise MFCC

	Block-wise MFCC			Fused features	
Genres	Recall	Precision	F_1-measure	One-shot Voting: Recall	Score Voting: Recall
Blues	0.97	0.97	0.98	1.00	1.00
Classical	0.98	0.97	0.98	1.00	1.00
Country	0.96	0.94	0.95	1.00	1.00
Disco	0.96	0.97	0.96	0.99	1.00
HipHop	0.95	0.96	0.96	0.98	0.98
Jazz	0.97	0.97	0.97	1.00	1.00
Metal	0.97	0.96	0.96	0.99	0.99
Pop	0.93	0.97	0.95	1.00	1.00
Reggae	0.96	0.95	0.95	1.00	1.00
Rock	0.95	0.94	0.95	1.00	0.99
Avg. (%)	96.0			99.7	99.8

Table 2. Overall results: recall rates (%) on the GTZAN music-genre classification problem, from [17] and completed with our results (in bold)

N^o	Classifier	Type of features	Recall
1	**Perceptrons**	**Learnt using DBN on MFCC**	**99.8**
2	CSC [15]	Many features	92.70
3	SRC [16]	Auditory cortical features	92
4	RBF-SVM [10]	Learnt using DBN on spectrum	84.3
5	Linear SVM [17]	Learnt using PSD on octaves	83.4
6	AdaBoost [4]	Many features	83
7	Linear SVM [17]	Learnt using PSD on frames	79.4
8	SVM [18]	Daubechies-Wavelets	78.5
9	Log. Reg. [19]	Spectral Covariance	77
10	LDA [14]	MFCC + other	71
11	Linear SVM [16]	Auditory cortical features	70
12	GMM [20]	MFCC + other	61

4 Conclusion

In this paper, we have proposed a new approach for musical genre classification which relies on learning a feature hierarchy over block-wise MFCC and outperforms the experiments that have been conducted until now. There are several reasons that may explain such an improvement of the state-of-the-art results. First, our strategy is based on MFCC that we tuned and customized to achieve temporal and timbral extraction. We then produce a set of different features which are of high-level and task specific thanks to the fine-tuning step. We also make use of sparsity because it is probably a privileged process to be able to learn such a number of features via the so called over-complete sparse representations. Concerning our classifier, we use a simple Single Layer Perceptron which could be "compared" to a linear-SVM used in most methods but that can refine the features previously found when fine-tuning with the back-propagation supervised learning. Unlike most approaches where each component may be trained (or fixed) "independently", our system is trained in a holistic way: each module influences the other ones during the different learning phases, leading to a powerful solution.

References

1. Tzanetakis, G.: Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing 10(5), 293–302 (2002)
2. Lidy, T., Rauber, A.: Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In: International Society for Music Information Retrieval Conference, pp. 34–41 (2005)
3. Tsuji, Y., Akahori, K., Nishikata, A.: The estimation of music genre using neural network and its educational use. In: International Conference on Computer-Assisted Instruction, pp. 158–162 (2000)
4. Bergstra, J., Kgl, B.: Aggregate features and adaboost for music classification. Machine Learning 2(65), 473–484 (2006)
5. Seyerlehner, K., Schedl, M., Pohle, T., Knees, P.: Using block-level features for genre classification, tag, classification and music similarity estimation. In: IMEX (2010)
6. Costa, Y., Oliveira, L., Koerich, A., Gouyon, F.: Music genre recognition using spectograms. In: WSSIP 2010, pp. 151–154 (2010)
7. Hua, B., Fu-long, M., Li-cheng, J.: Research on computation of glcm of image texture (2006)
8. Li, T.L., Chan, A., Chun, A.: Automatic musical pattern feature extraction using convolutional neural network. In: IMECS 2010 (2010)
9. Hinton, G.: To recognize shapes, first learn to generate images. Progress in Brain Research 165, 535–547 (2006)
10. Hamel, P., Eck, D.: Learning features from music audio with deep belief networks. In: International Society for Music Information Retrieval, pp. 339–344 (2010)
11. Ranzato, M., Boureau, Y.-L., Chopra, S., Lecun, Y.: A unified energy-based framework for unsupervised learning. Journal of Machine Learning Research 2, 371–379 (2007)
12. Bridle, J., Brown, M.: An experimental word recognition system, jsru report no 1003. Joint Speech Research Unit, Ruislip, England, Tech. Rep. (1974)
13. Li, T.L., Chan, A.: Genre classification and the invariance of mfcc features to key and tempo. In: International Conference on MultiMedia Modeling (2011)
14. Li, T.L., Tzanetakis, G.: Factors in automatic musical genre classification. In: IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (2003)
15. Chang, K., Jang, J., Ilioupoulos, C.: Music genre classification via compressive sampling. In: International Society for Music Information Retrieval, pp. 387–392 (2010)
16. Panagakis, Y., Kotropoulos, C., Arce, G.: Music genre classification using locality preserving non-negative tensor factorization and sparse representations. In: International Society for Music Information Retrieval, pp. 249–254 (2009)
17. Henaff, M., Jarett, K., Kavukcuoglu, K., LeCun, Y.: Unsupervised learning of sparse features for scalable audio classification. In: International Society for Music Information Retrieval (2011)
18. Li, T.L., Ogihara, M., Li, Q.: A comparative study on content-based music genre classification. In: ACM SIGIR Conference on Research and Development in Information Retrieval (2003)
19. Bergstra, J., Mandel, M., Eck, D.: Scalable genre and tag prediction using spectral covariance. In: International Society for Music Information Retrieval (2010)
20. Smith, E., Lewicki, M.: Efficient auditory coding. Nature (2006)

Exploration of Loneliness Questionnaires Using the Self-Organising Map

Krista Lagus[1], Juho Saari[2], Ilari T. Nieminen[1], and Timo Honkela[1]

[1] Aalto University School of Science, Dep't of Information and Computer Science
P.O. Box 15400, FI-00076 Aalto, Finland

[2] University of Eastern Finland, Dep't of Social Sciences
P.O. Box 1627, FI-70211 Kuopio, Finland

Abstract. Statistical machine learning methods can provide help when developing preventative services and tools that support the empowerment of individuals. We explore how the self-organizing map could be utilized as a tool for analyzing, visualizing and browsing heterogeneous survey data on wellbeing that contains both quantitative (numeric) and qualitative (text) data. There is systematic evidence implying that social isolation has drastic consequences for subjective well-being and health. It is important to obtain a deeper understanding of the phenomenon. Analysis of loneliness questionnaire data (N=521) succeeds in identifying profiles of loneliness as well as identifies crowd-sourced ideas for improving social wellbeing among the different subgroups.

1 Introduction

Statistical machine learning methods are increasingly applied in understanding complex questions related to human and social sciences (see, e.g., [1]). These phenomena are often very dynamic and involve a large number of variables with a substantial number of feedback loops. Therefore, the use of traditional quantitative methods like hypothesis-driven statistical analysis may lead to overly simplified conclusions. When there is a large number of variables involved or when text data is analyzed along with numerical data, dimensionality reduction and visualization methods are useful. The self-organizing map (SOM) [2] is one of the most popular data visualization methods. It is particularly useful hen the faithfulness of the mapping from a high-dimensional space into a low-dimensional space is important [3]. The SOM is widely used for data analysis and visualization, i.e., in biosciences [4], while relatively unknown within social sciences and humanities. In the latter, it has substantial potential in helping to deal with complex phenomena and in linking quantitative and qualitative research (see, e.g., [5,6]).

1.1 Support for Wellbeing Using Web Services and Data Mining

Research on human health is becoming an increasingly multidisciplinary and interdisciplinary endeavor. In addition to the traditional view on health as a biological and medical phenomenon, its cognitive, psychological, social and societal dimensions have been acknowledged as well. A sign of this kind of broadening of research focus is the

© Springer-Verlag Berlin Heidelberg 2013

use of the term “wellbeing” instead of the term “health”. Human health and wellbeing is a dynamic phenomenon that is influenced by a number of physical, physiological, psychological and social variables such as age, weight, education, family and social relations, attitudes and habits [7].

The objective of our project called VirtualCoach has been to explore the development of preventative services and tools that support the empowerment of individuals. Our aim has been to apply statistical data analysis and machine learning methods and tools in the area of wellbeing research. This research area can be called *wellbeing informatics*. In practical applications, the methodological basis is closely intertwined with web-based services and crowdsourcing. Wellbeing informatics can then be defined as the activity where observations regarding individual wellbeing are collected and analysed using methods suitable for finding value in large data sets. We define the final purpose of wellbeing informatics as to identify and share socially such information to relevant other individuals [8].

Earlier in our wellbeing informatics research, we have analyzed fitness data [9,8], compared different visualization methods such as multidimensional scaling (MDS), self-organising map, generative topographic mapping (GTM) and neighbour retrieval visualiser (NeRV) in relation to stress data [10], developed means for finding wellbeing-related stories that are relevant regarding their topic and suitable regarding their emotive content [11], and studied how to analyze and visualize trajectories of the development of wellbeing [8].

2 Loneliness Questionnaires as an Object of Study

In our research, the main underlying questions have been the following. (1) How to conduct quantitative and qualitative analysis on survey data related to loneliness? (2) Can the analysis uncover useful and non-trivial findings from experiences or reports of loneliness? Can we detect paths leading to loneliness, or escape routes? (3) Is it possible to transfer respectful encounters into a virtual forum? Can questionnaires serve as an intervention? In this article, we focus on the first two questions and discuss the third in the conclusions. In the following, we discuss loneliness as an object of study, describe the collected data and the method used in the analysis.

Most individuals occasionally feel socially isolated. However, for some such a feeling seems to be a quite permanent state of emotional affairs. Most surveys imply that the proportion of permanently isolated varies between 10 - 20 % depending on the ways of wording the surveys. Sometimes it is simply asked using different scales and time frames, whether the responded feels lonely; in some other times, it is asked whether the responded has somebody with whom they can share the most private issues. As a rule, one finds that the stronger the rate of isolation, the higher the identity of lonely person.

While in some countries, there is some evidence on the increasing proportion of population experiencing loneliness, there is no universal evidence on such a trend, contrary to the opinion of many social commentators who argue the opposite on the basis of sporadic evidence. Time series on the issue are scarce and further evidence is clearly needed. Furthermore, the proportion of lonely persons seems to be the lowest in small welfare states whereas the highest proportions of loneliness are found in transition economies, like Bulgaria.

Regardless of its proportion, there is some systematic evidence implying that social isolation has drastic consequences for subjective well-being (SWB) and health (SH). In both cases, the co-efficient of variation is quite strong and universally statistically significant. Furthermore, causally, there is some systematic evidence that isolation results in numerous unwanted social and mental states.

It remains somewhat unclear, however, why social isolation has so drastic consequences for SWB and SH. There are some tentative theoretical frames. An evolutionary framework argues in favor of social animal hypothesis and argues that despite of selfish genes we are co-operative animals who benefit from social interaction. Therefore, the emotional pain resulting from social isolation has some evolutionary gains. Sociologically, social isolation reflects the degree of social solidarity in a society, more precisely, the lack of it; in this context, social isolation is a way to measure anomie in a society.

Social isolation or loneliness can be defined as the subjective evaluation that the number of relationships is smaller than the individual considers desirable or that the intimacy that the individual wishes for has not been realized [12]. Social isolation is a severe health risk both physically and mentally [13]. Those who suffer from loneliness most include people who are ill, unemployed, or who are single parents [14]. Regarding children, it has been found out that cohesive families exhibit the lowest levels of loneliness and the highest levels of personal strengths [15]. There is a link between loneliness and depression. Rumination mediates the relationship between peer-related loneliness and depressive symptoms and moderates the relationship between parent-related loneliness and depressive symptoms [16]. Rumination refers to the habit of focusing on a repetitive manner on symptoms of distress and on the possible causes and consequences of these symptoms. When parents feel lonely, there are also consequences for the children. Parental loneliness and a history of being bullied are each significant predictors of young adult loneliness [17]. On the other hand, a family environment that supports open communication is negatively correlated with loneliness of young adults [17]. It has been found out that in the southern and central European countries, loneliness is largely attributable to not being married, economic deprivation, and poor health [12].

2.1 Data Collection

We conducted a survey on loneliness in association with the Common Responsibility Campaign (in Finnish "yhteisvastuukeräys", see the web site yhteisvastuu.fi/en/ for details). The questionnaire was advertised at the web page of the campaign for 2.5 months during Spring 2011. It was targeted to people who are lonely now or have sometimes experienced loneliness, and obtained 521 answers. Questionnaire design took place in the framework of positive psychology [18] with the goal of extracting profiles and paths using data analysis. The questionnaire included five open (text) questions, 23 closed questions with Likert scale, a question on the loneliness type, and background questions related to the age, gender, and the geographical area and size of place of living. The open questions were: (1) How did you become lonely? Please describe the circumstances. (2) How did it feel? How did it affect your mind and behaviour? (3) How did you survive loneliness (or despite it)? Where did you find resources? What kind of survival mechanisms did you develop? (4) Did people close to you know about your loneliness? How did they react? (5) What would you like to say to others in a similar situation?

2.2 Methods

We used the self-organizing map (SOM) [2] to analyze the questionnaire data. Within human and social sciences, the SOM has been used, for instance, in the analysis of voting patterns [19], religious conceptions [20], large text collections [21], and conceptual similarities of words [22].

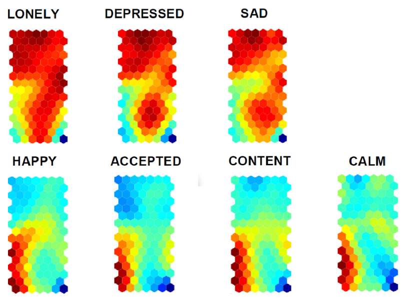


Fig. 1. Condition in the worst year of life. Dark red marks the highest value for a variable. Strong correlations are visible. Those worst off are found in the upmost part.

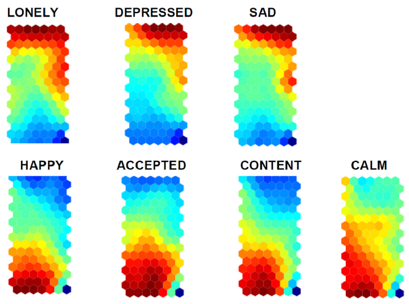


Fig. 2. Condition during last month. The individuals with largest positive changes are found in the middle and lower part of the map.

3 Results

Closed-class question data was analyzed using the SOM [2]. Each of the 521 individuals was treated as a data point, represented as a vector. The 23-dimensional vectors were normalized, and ordered automatically on the two-dimensional lattice by the SOM algorithm using SOM ToolBox for MATLAB[1]. A visualized map interface was developed for browsing the text answers to facilitate qualitative analysis.

Figures 1 and 2 indicate the development of the persons' condition from their worst year to last month. There is a clear general positive tendency. The area related to negative conditions (loneliness, depression and sadness) has decreased significantly and given room to positive conditions (being happy, accepted, content and calm). As expected, the positive as well as negative conditions correlate in the group to a large degree. These first maps give rise to two important questions: (1) What can be said about people whose condition has improved, and (2) What is characteristic of people whose condition has remained problematic? The first question can be examined from by Figure 3 that indicates level of content regarding various aspects of life, and helps identify the loneliness profiles that are then pointed out explicitly in Figure 4. Regarding the second question, Figure 5 depicts an overview of the means of surviving loneliness that the written responses in the "survivors" group revealed.

[1] Euclidean geometry was applied for distance calculation since it was available in the tool, however with categorical variables one might wish to consider other metrics [23].

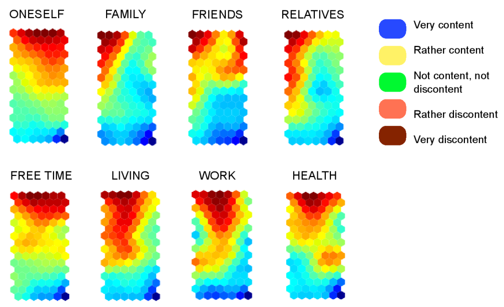


Fig. 3. Levels of being content with different aspects of life. Here different sub-groups in the "most lonely" upper part of the map can be identified. In particular, upper right corner contains individuals who are delatively content with their family, relatives whereas upper left group is discontent about all relationships.

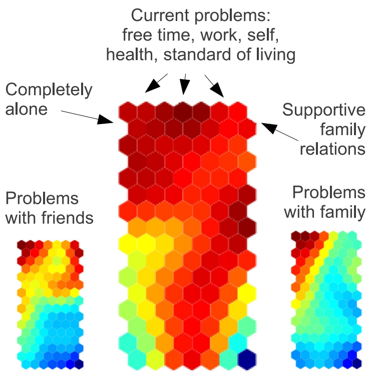


Fig. 4. Profiles of loneliness among the chronically lonely

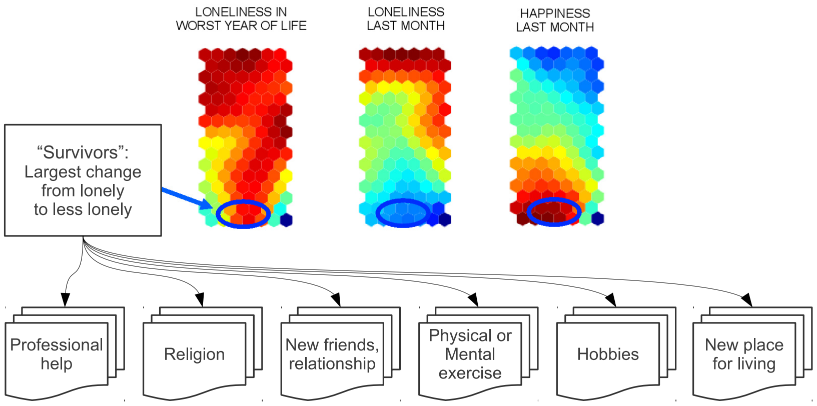

LONELINESS IN WORST YEAR OF LIFE
LONELINESS LAST MONTH
HAPPINESS LAST MONTH
"Survivors": Largest change from lonely to less lonely
Professional help
Religion
New friends, relationship
Physical or Mental exercise
Hobbies
New place for living

Fig. 5. Topics that the "survivors" related to positive developments

4 Conclusions and Discussion

From a methodological perspective, we show how the SOM or similar visualization principle can be useful in the analysis of rather typical social sciences data - namely questionnaires containing both closed-class questions and open text questions. The map allows looking at correlations among several variables at once in a way that is richer than correlations table, but less precise. The visualization allows the researcher to gain an overview of the phenomenon and subsequently focus reading of the written accounts to the most interesting part of the data set. The benefit is that the researcher can collect a practically unrestricted amount of data, and still make useful conclusions about it.

On social isolation we can conclude that loneliness, depression and sadness are highly correlated. An interesting finding are the different loneliness profiles based on the different types of support that people experience and lean to, as well as the identification of the group of survivors. The latter group offers hope and crowdsourced ideas on how to survive loneliness. Future applications include the possibility to design interventions targeted to the different loneliness profiles. For example, the group in the upmost part of the map appears to require more intensive professional help possibly in several areas of health and social services. In contrast, the group in the right who are extremely lonely but relatively happy with work and standard of living etc, might benefit from a different type of support, such as social activities organized in the work context, or ideas obtained from the survivors group.

The questionnaire was designed to be a positive intervention in the lives of the respondents. Based on the unexpectedly large number of answers as well as the written feedback, we tentatively conclude that questionnaires when designed from a respectful encounters perspective can indeed be positive interventions. This is extremely important from an ethical perspective, especially when researching individuals who are already in a difficult situation and whose resources may be low - so as to not tax their resources further, but instead to empower the individuals. We have also applied our approach with success in the context of collecting data regarding stress and recovery, as well as on studying breastfeeding experiences.

References

1. Castellani, B., Hafferty, F.: Sociology and Complexity Science: A New Field of Inquiry. Springer (2009)
2. Kohonen, T.: Self-organizing maps. Springer (2001)
3. Venna, J., Kaski, S.: Local multidimensional scaling. Neural Networks 19(6), 889–899 (2006)
4. Nikkilä, J., Törönen, P., Kaski, S., Venna, J., Castrén, E., Wong, G.: Analysis and visualization of gene expression data using self-organizing maps. Neural networks 15(8), 953–966 (2002)
5. Castellani, B., Castellani, J., Spray, S.L.: Grounded neural networking: Modeling complex quantitative data. Symbolic Interaction 26(4), 577–589 (2003)
6. Janasik, N., Honkela, T., Bruun, H.: Text mining in qualitative research: Application of an unsupervised learning method. Organizational Research Methods 12(3), 436–460 (2009)

7. Honkela, T., Koskinen, I., Koskenniemi, T., Karvonen, S.: Kohonen's Self-Organizing Map in Contextual Analysis of Data. In: Information Organization and Databases: Foundations of Data Organization, pp. 135–148. Kluwer (2000)
8. Lagus, K., Vatanen, T., Kettunen, O., Heikkilä, A., Heikkilä, M., Pantzar, M., Honkela, T.: Paths of wellbeing on self-organizing maps. In: Proc. of WSOM 2012, pp. 345–352 (2012)
9. Vatanen, T., Heikkilä, M., Honkela, T., Kettunen, O., Lagus, K., Pantzar, M.: Kuntotiedot kartalle - erilaiset hyvä- ja huonokuntoisten ryhmät näkyviin. Liikunta & Tiede (Sports & Science), 48–53 (2012)
10. Heikkilä, A.: Information visualisation in a peer support application. Master's thesis, Aalto University, Department of Information and Computer Science, Espoo, Finland (2012)
11. Honkela, T., Izzatdust, Z., Lagus, K.: Text mining for wellbeing: Selecting stories using semantic and pragmatic features. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 467–474. Springer, Heidelberg (2012)
12. Fokkema, T., De Jong Gierveld, J., Dykstra, P.A.: Cross-national differences in older adult loneliness. The Journal of psychology 146(1-2), 201–228 (2012)
13. Hagerty, B.M., Williams, A.: The effects of sense of belonging, social support, conflict, and loneliness on depression. Nursing Research 48(4), 215–219 (1999)
14. Saari, J.: Yksinäisten yhteiskunta (The society of the lonely). WSOY (2009)
15. Sharabi, A., Levi, U., Margalit, M.: Children's loneliness, sense of coherence, family climate and hope: Developmental risk and protective factors. The Journal of Psychology 146(1-2), 61–83 (2012)
16. Vanhalst, J., Luyckx, K., Raes, F., Goossens, L.: Loneliness and depressive symptoms: The mediating and moderating role of uncontrollable ruminative thoughts. The Journal of psychology 146(1-2), 259–276 (2012)
17. Segrin, C., Nevarez, N., Arroyo, A., Harwood, J.: Family of origin environment and adolescent bullying predict young adult loneliness. The Journal of Psychology 146(1-2), 119–134 (2012)
18. Seligman, M.E.: Positive psychology, positive prevention, and positive therapy. Handbook of Positive Psychology 2, 3–12 (2002)
19. Pearson, P.T., Cooper, C.I.: Using self organizing maps to analyze demographics and swing state voting in the 2008 U.S. Presidential election. In: Mana, N., Schwenker, F., Trentin, E. (eds.) ANNPR 2012. LNCS, vol. 7477, pp. 201–212. Springer, Heidelberg (2012)
20. Pyysiäinen, I., Lindeman, M., Honkela, T.: Counterintuitiveness as the hallmark of religiosity. Religion 33(4), 341–355 (2003)
21. Kaski, S., Honkela, T., Lagus, K., Kohonen, T.: WEBSOM—self-organizing maps of document collections. Neurocomputing 21, 101–117 (1998)
22. Lagus, K., Airola, A., Creutz, M.: Data analysis of conceptual similarities of Finnish verbs. In: Proceedings of CogSci 2002, pp. 566–571 (2002)
23. Cottrell, M., Ibbou, S., Letremy, P.: Som-based algorithms for qualitative variables. Neural Networks 17(8-9), 1149–1167 (2004)

An Effective Dynamic Gesture Recognition System Based on the Feature Vector Reduction for SURF and LCS

Pablo V.A. Barros, Nestor T.M. Júnior, Juvenal M.M. Bisneto,
Bruno J.T. Fernandes, Byron L.D. Bezerra, and Sérgio M.M. Fernandes

Polytechnics School - University of Pernambuco - Recife, Brazil
{pvab,ntmj,jmmb,bjtf,byronleite,smmf}@ecomp.poli.br

Abstract. Speed Up Robust Feature (SURF) and Local Contour Sequence(LCS) are methods used for feature extraction techniques for dynamic gesture recognition. A problem presented by these techniques is the large amount of data in the output vector which difficult the classification task. This paper presents a novel method for dimensionality reduction of the features extracted by SURF and LCS, called Convexity Approach. The proposed method is evaluated in a gesture recognition task and improves the recognition rate of LCS while SURF while decreases the amount of data in the output vector.

1 Introduction

Hand gesture recognition system provides a natural, user-friendly way of interaction with the computer which is more familiar to the human beings. Gesture recognition has a wide area of application including human machine interaction, sign language and video game technology. Most of the dynamic gesture recognition systems are divided in two steps: feature extraction and pattern classification. Several techniques are applied for feature extraction but many of them might fall in dimensionality curse as show in the survey published by Bilal et al[3]. The dimensionality curse says that the numerical approximation of a function will require more computation as the number of active variables grows [9].

Meena [11] uses the Local Contour Sequence (LCS) algorithm to calculate the feature vector for gesture recognition. This technique generates an output with many features by the calculation of the distance of all the points that compound a hand posture, which can difficult the learning process. Bao et al. [1] use the Speed Up Robust Features (SURF) to recognize gestures trough tracking. In their work, they use the SURF points in adjacent frames to help describing a hand trajectory. SURF generates many points, as it is applied in all the image an returns the interest points based in different image transformations and not in the observed object. Jiang et al. [8] describe full body gestures using SURF and Bag of Video Worlds Model to normalize the interest points extracted by SURF. This method uses a set of SURF descriptors for each interest point

© Springer-Verlag Berlin Heidelberg 2013

extracted. This descriptors are obtained summing the HAAR wavelets response around the interest points and thus generates a large feature vector, containing each description vector for all the interest points. Yao et al. [15] use SURF to extract the key points of a hand posture and uses Adaboost to decrease the computational cost for the training. Yao et al. method increase the classification efficiency by using a hybrid classification model, but the feature vector still contain the same amount of features, and it could be minimized.

This paper introduces a novel method to reduce the feature vector size. The Convexity approach extends LCS and SURF and shows a better result in classification of dynamic gestures. A experiment in gesture recognition is performed and demonstrate the efficiency and efficacy of the approach with three different pattern classification methods: Dynamic Time Wrapper [13], Hidden Markov Model [12] and Elman Recurrent Neural Network [7].

This paper is structured as follows: Section II describes the Convexity Approach algorithm. Section III presents the experimental results. Finally, in Section IV, the conclusions and some future work are given.

2 Convexity Approach

The Convexity Approach reduces a feature vector, choosing the smallest group of points that can represent the hand posture. This algorithm is applied in Local Contour Sequence (LCS), created by Gupta [4], and in Speed Up Robust Features (SURF) described by Bay [2] . The input for the Convexity Approach is a set of points that represents the hand posture. The first step of the Convexity Approach algorithm is to minimize the hand posture. The second step is to find the convex hull of the previously selected points. The last step uses the points that composes the convex hull for a feature calculation based on point distance. Figure 1 shows the execution illustration of the Convexity Approach.

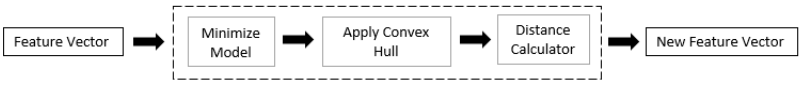
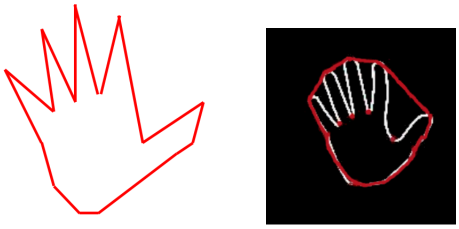


Fig. 1. Convexity Approach execution illustration

The Douglas-Peucker algorithm [6] is used in the first step to create an approximation curve of the external points, forming a minimized polygon for hand posture. In this algorithm, the two extreme endpoints of a set of points are connected with a straight line as the initial rough approximation of the polygon. Then, it approximates the whole polygon by computing the distance from all intermediate polygons vertices to that line segment. If all these distances are less than the specified tolerance T, then the approximation is good, the endpoints are retained, and the other vertices are eliminated. However, if any of these distances exceeds the T tolerance, then the approximation is not good enough.



(a) Polygon Minimization (b) Convex Hull and inner points selection

Fig. 2. Outputs of step one (a) and two (b) of Convexity Approach

In this case, it chooses the point that is furthest away as a new vertex subdividing the original set points into two set points. This procedure is repeated recursively on these two shorter set points. If at any time, all of the intermediate distances are less than the T threshold, then all the intermediate points are eliminated. The routine continues until all possible points have been eliminated. Figure 2(a) shows the output of this step.

The second step is to select the most significant points for the specifically hand posture. We run the Sklankys [14] algorithm in the last step output.The algorithm consists in the following sequence:

- The convex vertex of the polygon is found.
- The remaining n-1 vertexes are named in clockwise order starting at P0.
- Select P0, P1 and P2 vertices and call then "Back", "Center" and "Front" respectively
- Execute the follow algorithm:

```
• while "Front" is not on vertex P0 and "Back", "Center" and "Front" form a right turn do
      if "Back", "Center" and "Front" form a left turn or are collinear vertex then
          change "Back" to the vertex ahead of "Front". Relabel "Back" to "Front", "Front" to "Center" and "Center" to "Back".
      else if "Back", "Center" and "Front" turn left then
          change "Center" to the vertex behind "Back", Remove the vertex and associated edges that "Center" was on and relabel "Center" to "Back" and "Back" to "Center"
      end if
  end while
```

- For each pair of selected points the algorithm traces a line. The most distant point of this line is selected as an inner point. Figure 2(b) shows the resultant points of the algorithm in a input image.

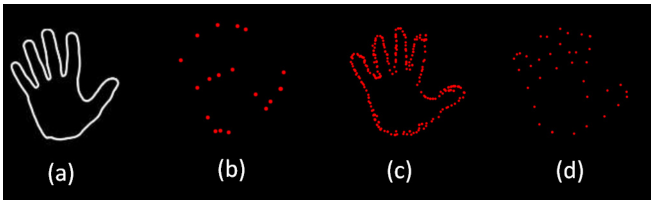


Fig. 3. Applying the Convexity Approach on LCS (a) produces a new output vector (b). Applying the Convexity Approach on SURF (c) produces a new output vector (d).

The last step of convexity approach is the feature extraction based on distance calculation. A line is formed by each pair of the external points chosen by step two. The distance between this line and the closer inner point is calculated and added to the output vector. Figure 3 shows the result obtained by LCS and SURF and the application of Convexity Approach, respectively CLCS and CSURF.

For some classification techniques, such as Neural Networks, the feature vector must be normalized. To solve this problem we propose a method. First, the number of normalized distances is defined. Then if the image has fewer points than the determined one, "0" are added in the feature vector until matches the desired length. The outputs with more points than the desired length are normalized using a selection algorithm. This algorithm consists in calculation of a window, W, through the division of the output length for the desired length. The output vector is traversed, and each W position is added to the new vector of outputs. If the new output vector is smaller than the desired length, the remaining positions are randomly visited and used to compose the new output vector until the desired length is achieved.

3 Dynamic Gesture Recognition

The efficiency and effectiveness of the Convexity Approach applied to LCS and SURF algorithms is evaluated in this task. We compare the results of the classification of dynamic gestures using the original version of the LCS and SURF with the extended one applying the Convexity Approach, called CLCS and CSURF, respectively.

3.1 Methodological Protocol

The database used for this test is the RPPDI Gesture Database[1]. It contains a set of four different gestures: Click, Grasp, No and Goodbye.

To test the algorithms a dynamic gesture recognition system is used and it is composed by two modules: extraction and classification. Four methods for feature extraction are evaluated: SURF, LCS, CSURF and CLCS. Three classification techniques are build: Dynamic Time Wrapper, Hidden Markov Model

[1] Available at `http://rppdi.ecomp.poli.br/gesture/database/`

Table 1. Best Techniques Configuration Combination with and without Convexity Approach

Technique	Parameters	CLCS	CSURF	LCS	SURF
Elman RNN	Neurons Input Layer	140	210	5600	1400
	Hidden Layers	12	14	75	37
	Neurons Output Layer	4	4	4	4
	Execution Average Time	21.843ms	27.953ms	72.662ms	76.543ms
	Taining Average Time	210824ms	180351ms	400298ms	550221ms
HMM	States	3	3	3	3
	Baum-Welch Iterations	100	100	10	10
	Features	10	10	5600	1400
	Execution Average Time	181.959ms	276.724ms	-	-
	Taining Average Time	0.446ms	0.690ms	-	-
DTW	Features	140	210	5600	1400
	Execution Average Time	45.743ms	164.964ms	640.713ms	1722.289ms
	Taining Average Time	129.015ms	133.620ms	156.048ms	208.191ms

and an Elman Recurrent Neural Network. The experiments are evaluated with the combination of each extraction technique with each classification technique.

The Recurrent Neural Network chosen for the classification module is the Elman Recurrent Neural Network(Elman RNN). The Backprogation with Simulated Annealing training strategy is used as it showed good results for dynamic sequences training in the work of Zhang [16]. The Hidden Markov Model technique uses a K-Means Clustering [5] to find the best initial approximation. The Baum-Welch algorithm [10] is used to train the HMM, resulting in a fast training process.

All experiments are repeated 30 times in a database division containing 2/3 of the sequences in each gesture class for training and 1/3 for testing, randomly chosen. The results present the mean among all repetitions. The best configuration results are showed in Table 1.

3.2 Experimental Results

The 5600 features of the LCS are too excessive to calculate the HMM probabilities, and it could not converge in a result. The same happens when using SURF. The HMM used with CLCS and CSURF reached a recognition rate of 91%, for both, as showed in Table 2.

Elman Recurrent Neural Network recognized 77% of the gestures using the LCS. Using SURF the recognition rate dropped to 72%. The CLCS recognition rate was 90% and the CSURF recognition rate was 92%. Table 3 shows the confusion matrix of all the extraction techniques combined with the RNN.

Table 2. Confusion Matrix using CLCS and CSURF as feature extraction technique techniques and HMM as classification technique

	CLCS					CSURF			
	Gest. 1	Gest 2.	Gest. 3	Gest. 4		Gest. 1	Gest 2.	Gest. 3	Gest. 4
Gest. 1	0	1	4	0	Gest. 1	1	0	4	0
Gest. 2	1	6	1	0	Gest. 2	0	8	0	0
Gest. 3	0	0	11	0	Gest. 3	2	0	9	0
Gest. 4	0	0	0	6	Gest. 4	0	0	0	6

Table 3. Confusion Matrix using LCS, CLCS, SURF and CSURF as feature extraction technique techniques and RNN as classification technique

	LCS					CLCS			
	Gest. 1	Gest 2.	Gest. 3	Gest. 4		Gest. 1	Gest 2.	Gest. 3	Gest. 4
Gest. 1	4	3	1	0	Gest. 1	7	1	0	0
Gest. 2	2	6	0	0	Gest. 2	1	7	0	0
Gest. 3	2	0	9	0	Gest. 3	1	0	10	0
Gest. 4	0	0	0	6	Gest. 4	0	0	0	6
	SURF					CSURF			
	Gest. 1	Gest 2.	Gest. 3	Gest. 4		Gest. 1	Gest 2.	Gest. 3	Gest. 4
Gest. 1	4	3	1	0	Gest. 1	7	0	1	0
Gest. 2	2	6	0	0	Gest. 2	0	8	0	0
Gest. 3	3	0	8	0	Gest. 3	2	0	9	0
Gest. 4	0	0	0	6	Gest. 4	0	0	0	6

Table 4. Confusion Matrix using LCS, CLCS, SURF and CSURF as feature extraction technique techniques and DTW as classification technique

	LCS					CLCS			
	Gest. 1	Gest 2.	Gest. 3	Gest. 4		Gest. 1	Gest 2.	Gest. 3	Gest. 4
Gest. 1	12	3	0	0	Gest. 1	13	0	2	0
Gest. 2	2	11	0	3	Gest. 2	2	16	0	0
Gest. 3	5	0	14	2	Gest. 3	2	0	21	0
Gest. 4	0	0	0	12	Gest. 4	0	0	0	12
	SURF					CSURF			
	Gest. 1	Gest 2.	Gest. 3	Gest. 4		Gest. 1	Gest 2.	Gest. 3	Gest. 4
Gest. 1	12	3	0	0	Gest. 1	13	2	0	0
Gest. 2	2	11	0	3	Gest. 2	1	15	1	0
Gest. 3	5	0	13	3	Gest. 3	1	20	0	0
Gest. 4	0	0	0	12	Gest. 4	0	0	0	12

Table 5. Classication rate resume for all the Feature Extraction/Classification techniques combinations

Combination	Clas(%)	Combination	Clas(%)	Combination	Clas(%)
HMM + LCS	-	RNN + LCS	77.0	DTW + LCS	77.0
HMM + SURF	-	RNN + SURF	72.0	DTW + SURF	75.0
HMM + CLCS	91.0	RNN + CLCS	90.0	DTW + CLCS	97.0
HMM + CSURF	91.0	RNN + CSURF	92.0	DTW + CSURF	93.0

Using DTW to classify the distances between the sequences provided by the LCS technique generated a recognition rate of 77%. Using the SURF sequences 75% of the gestures were recognized correctly. Using CLCS the recognition rate was 97% and CSURF got 93% of the correct gestures recognized. Table 4 shows the confusion matrix.

Table 5 shows the resume of this session. The combinations of feature extraction techniques with classification techniques was evaluated. Except for LCS and SURF in HMM because the amount of data was too excessive, a problem corrected by Convexity Approach. It shows that the Convexity Approach obtained a higher classification rate in all the combinations. The results shows the use of Convexity Approach decreased the size of extracted feature vector and also increased the recognition rate for all the classification techniques. It shows also that the Convexity Approach lowered the classification and execution time for all techniques.

4 Conclusion

The Convexity Approach creates a minimized feature vector for dynamic gesture recognition. It uses a dynamic selection of points, based in the point significance in the hand posture model, to reduce the total point given by a feature extraction technique.

To evaluate the Convexity Approach in a gesture recognition task, it is used in LCS and SURF, and the results used in two in two dynamic gesture recognition system, one based on HMM and one based on DTW. The results are compared and showed that Convexity Approach got better recognition rates.

The Future Works can be listed as: Apply the Convexity Approach to others gesture feature extraction techniques and test the CLCS and CSURF in benchmark gesture recognition databases.

Acknowledgments. This work was partially supported by Brazilian agencies: CNPq, CAPES and FACEPE. This project is possible due the FACEPE process number APQ-0949-1.03/10 - "Reconhecimento de gestos uma aplicação para reconhecimento de sinais de surdos com dispositivos móveis".

References

1. Bao, J., Song, A., Guo, Y., Tang, H.: Dynamic hand gesture recognition based on surf tracking. In: 2011 International Conference on Electric Information and Control Engineering (ICEICE), pp. 338–341 (April 2011)
2. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. 110(3), 346–359 (2008), `http://dx.doi.org/10.1016/j.cviu.2007.09.014`
3. Bilal, S., Akmeliawati, R., El Salami, M., Shafie, A.: Vision-based hand posture detection and recognition for sign language; a study. In: 2011 4th International Conference on Mechatronics (ICOM), pp. 1–6 (2011)
4. Gupta, L., Ma, S.: Gesture-based interaction and communication: automated classification of hand gesture contours. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 31(1), 114–120 (2001)
5. Hartigan, J.A., Wong, M.A.: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28, 100–108 (1979)
6. Heckbert, P.S., Garland, M.: Survey of polygonal surface simplification algorithms (1997)
7. Jain, L.: Recurrent Neural Networks, 1st edn. CRC Press (2001)
8. Jiang, X., Sun, T., Feng, B., Jiang, C.: A space-time surf descriptor and its application to action recognition with video words. In: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 3, pp. 1911–1915 (July 2011)
9. Kouiroukidis, N., Evangelidis, G.: The effects of dimensionality curse in high dimensional knn search. In: 2011 15th Panhellenic Conference on Informatics (PCI), pp. 41–45 (2011)
10. Baum, L., Peterie, T., Souled, G., Weiss, N.: A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. In: Proceedings of Annals of Mathematical Statistics, vol. 41, pp. 164–171 (1995)
11. Meena, S.: A Study on Hand Gesture Recognition Technique. Master's thesis, National Institute of Technology, Rourkela,India (2011)
12. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. In: Readings in Speech Recognition, pp. 267–296. Morgan Kaufmann Publishers Inc., San Francisco (1990), `http://dl.acm.org/citation.cfm?id=108235.108253`
13. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. In: Readings in Speech Recognition, pp. 159–165. Morgan Kaufmann Publishers Inc., San Francisco (1990), `http://dl.acm.org/citation.cfm?id=108235.108244`
14. Sklansky, J.: Finding the convex hull of a simple polygon. Pattern Recogn. Lett. 1(2), 79–83 (1982), `http://dx.doi.org/10.1016/0167-8655(82)90016-2`
15. Yao, Y., Li, C.-T.: Hand posture recognition using surf with adaptive boosting. In: British Machine Vision Conference (2012)
16. Zhang, H., Wang, Y., Deng, C.: Application of gesture recognition based on simulated annealing bp neural network. In: 2011 International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), vol. 1, pp. 178–181 (August 2011)

Feature Weighting by Maximum Distance Minimization

Jens Hocke and Thomas Martinetz

University of Lübeck - Institute for Neuro- and Bioinformatics
Ratzeburger Allee 160 23538 Lübeck - Germany
hocke@inb.uni-luebeck.de

Abstract. The k-NN algorithm is still very popular due to its simplicity and the easy interpretability of the results. However, the often used Euclidean distance is an arbitrary choice for many datasets. It is arbitrary because often the data is described by measurements from different domains. Therefore, the Euclidean distance often leads to a bad classification rate of k-NN. By feature weighting the scaling of dimensions can be adapted and the classification performance can be significantly improved. We here present a simple linear programming based method for feature weighting, which in contrast to other feature weighting methods is robust to the initial scaling of the data dimensions. An evaluation is performed on real-world datasets from the UCI repository with comparison to other feature weighting algorithms and to Large Margin Nearest Neighbor Classification (LMNN) as a metric learning algorithm.

Keywords: feature selection, feature weighting, metric learning, k-Nearest-Neighbor, Relief, Large Margin Nearest Neighbor Classification.

1 Introduction

In pattern recognition tasks data is often described by measurements from different domains. Thus the feature dimensions of the datasets have an arbitrary scaling relative to each other. In many applications the k-Nearest-Neighbor classifier (k-NN) [1] is applied, because it is the simplest non-linear classifier and, which is even more important, the classification decisions are easily interpretable. The interpretability is due to the k nearest neighbors used for the classification decision. By inspecting these nearest neighbors one can discern the causes for the decision. However, the standard metric used for k-NN is the Euclidean metric, which does not measure the distance according to the relevance of each data dimension but according to their arbitrary scaling. To archive a performance close to state of the art classifiers, the dimensions need to be rescaled according to their relevance. A similar problem is solved by relevance learning in the context of LVQ classifiers [4].

An optimal rescaling has to minimize the classification error $E(X)$ of the k-NN algorithm. Often this problem is called the *feature weighting* problem. We want to find a weight vector $\boldsymbol{w} \in \Re^D, w_\mu \geq 0, \mu = 1, ..., D$ for some given dataset

© Springer-Verlag Berlin Heidelberg 2013

$X = \{\boldsymbol{x}_i \in \Re^D, i = 1, ..., N\}$ that helps the classifier to minimize $E(X)$. In case the Euclidean distance is used, the weighted distance between two data points $\boldsymbol{x}, \boldsymbol{x}'$ becomes

$$d(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{x} - \boldsymbol{x}'||_{\boldsymbol{w}} = \sqrt{\sum_{\mu=1}^{D} w_\mu (x_\mu - x'_\mu)^2}, \tag{1}$$

also called the weighted Euclidean distance. In case there are data dimensions irrelevant for the classification, the weights for these dimensions will be decreased to zero and, through this process, the dimensionality of the data set is reduced. Such a dimensionality reduction is desirable to increase generalization performance and noise robustness.

Several methods for feature weighting have been proposed. Very well known is the Relief algorithm by Kira and Rendell [5]. It iteratively updates a weight vector based on the distance of a data point to the nearest neighbors of the same and different label. This concept has been extended in the Simba algorithm [3] by incorporating the weight vector into the distance measure for finding the nearest neighbors. While both methods were developed for selecting the most important dimensions, they use a weight vector to do so. Further feature weighting methods are I-Relief [7], which also extends Relief, and the loss-margin based algorithm (Lmba) [6], which was derived from the Large Margin Nearest Neighbor Classification described in the next paragraph.

Feature weighting can be seen as a subclass of the *metric learning* problem. In metric learning often a Mahalanobis distance

$$d(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{x} - \boldsymbol{x}'||_W = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T W (\boldsymbol{x} - \boldsymbol{x}')} \tag{2}$$

is optimized to improve the k-NN classification. Note that here an entire positive semidefinite matrix W is optimized. The problem becomes equivalent to feature weighting, if W is restricted to a diagonal matrix. A well-known method for metric learning is Large Margin Nearest Neighbor Classification (LMNN) [8–11]. We will have a closer look at LMNN in the next section.

Here we present a simple linear programming approach for feature weighting. It archives a robust rescaling of the feature dimensions by a minimization of the maximum distance between data points of the same class under a minimal distance constraint for differently labeled points. We therefore name it Maximum Distance Minimization (MDM).

2 Methods

Before we introduce our approach, we first want to have a closer look at LMNN. Even though it is a metric learning method, whereas we present feature weighting, both methods are related by the goals they pursue to improve the k-NN classification performance.

2.1 LMNN

The LMNN algorithm is designed to optimize k-NN classification performance. To increase the classification performance, it pursues two goals directly derived from the k-NN. Since data points are classified by k-NN according to their k nearest neighbors, it would be best if for every class the data points from the same class are close together and points from different classes are far away. More precisely, the k closest points to every data point should have the same label. We name the k points $\boldsymbol{x}_j$ closest to point $\boldsymbol{x}_i$ and of the same class target neighbors. Any differently labeled point $\boldsymbol{x}_l$ which is closer than the most distant target neighbor (plus some margin) we name impostor. Now every triple of (i) data point, (ii) target neighbor and (iii) impostor is optimized. Additionally, the distance between all datapoints from the same class is minimized. The entire optimization can be done by the following positive-semidefinite program:

$$\min\ (1-\mu)\sum_{i,j\rightsquigarrow i}||\boldsymbol{x}_i-\boldsymbol{x}_j||_W^2+\mu\sum_{i,j\rightsquigarrow i,l}(1-y_{il})\xi_{i,j,l}\ \text{s.t.} \tag{3}$$

$$||\boldsymbol{x}_i-\boldsymbol{x}_l||_W^2-||\boldsymbol{x}_i-\boldsymbol{x}_j||_W^2\geq 1-\xi_{i,j,l} \tag{4}$$

$$\xi_{i,j,l}\geq 0 \tag{5}$$

$$W\succeq 0. \tag{6}$$

The notation $j \rightsquigarrow i$ means $\boldsymbol{x}_j$ is a target neighbor of $\boldsymbol{x}_i$. To indicate that $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ have the same label, $y_{i,l}$ is used, which equals one if that is the case and zero otherwise. μ is a parameter for weighting the two terms, and $\xi_{i,j,l}$ are the slack variables. By restricting the matrix W to a diagonal matrix, the optimization problem can be cast into a linear program.

There are a couple of problems with the above formulation. First, the target neighbors need to be chosen prior to the optimization. This is usually done based on the initial Euclidean distance. By this selection the final result of the optimization depends on the initial scaling of the dimensions. Second, there are many parameters to optimize due to the large number of slack variables. And third, there is a large number of constraints to keep track of.

2.2 MDM

Our Maximum Distance Minimization (MDM) is a feature weighting method. The objectives are not as closely linked to the concept of the k-NN as it is the case for LMNN, but it avoids the target neighbor selection problem and has much fewer parameters to optimize. With its very general objective it might also be well suited as a preprocessing step for other classifiers. Like LMNN we try to minimize the distance of data points of the same class. However, in our case we do not look at local neighbors but try to minimize the maximum distance between all pairs of data points of the same class, while keeping the pairwise distance between data points of different classes large. This means that we aim to get all data points of the same class as closely together as possible. This is a very global optimization. Formally, we are solving the following constrained optimization problem

$$||\boldsymbol{x}_i - \boldsymbol{x}_l||^2_{\boldsymbol{w}} \geq 1 \; \forall i,l : y_i \neq y_l \tag{7}$$
$$||\boldsymbol{x}_i - \boldsymbol{x}_j||^2_{\boldsymbol{w}} \leq r \; \forall i,j : y_i = y_j \tag{8}$$
$$\min_{\boldsymbol{w}} r \quad w_\mu \geq 0 \; \forall \mu, \tag{9}$$

where y_i, y_l, and y_j are the class labels of $\boldsymbol{x}_i$, $\boldsymbol{x}_l$, and $\boldsymbol{x}_j$.

The above problem can be formulated as a linear program[1]. For this formulation the number of constraints grows quadratically with the number of data points, but in contrast to LMNN we only have few parameters to optimize, namely only the weights w_μ. In contrast to LMNN, our optimization problem is always solvable, even without slack variables. While LMNN uses triples of points for optimization, here we only look at pairs. By summing the squared distances of target neighbors, LMNN uses a soft penalty for large distances. MDM has a hard penalty, which punishes only the most distant pairs. This may make MDM more sensible to outliers and noisy data. However, in our experiments we did not notice this problem. Of course softness can be added to MDM as well as target neighborhoods, but thereby some of the advantages of MDM would be lost.

3 Experiments

We evaluated MDM on datasets from the UCI repository [2]. MDM was compared with results based on standard Euclidean distance as well as obtained with the feature weighting algorithms Relief, Simba and the diagonally restricted LMNN (D-LMNN). As a reference, we also determined the result obtained with the unrestricted LMNN as a metric learning method. The parameter μ was chosen to be 0.5 for LMNN, because this was described by the authors to be a good choice. For D-LMNN μ was set to one, because for lower values it tended to reduce the dimensions too much. The datasets were split into 50% training points and 50% test points. After optimizing the weight vectors on the training data, the k-NN error rate was determined on the reweighted test data. Every algorithm was trained and tested on 10 different splits of every dataset. Table 1 shows the results obtained without any preprocessing of the data. In Table 2 the dataset dimensions were rescaled in a preprocessing step so that for every dimension the data points had a normalized distribution.

Without preprocessing (Table 1) MDM achieves the best error rates. It significantly improves the error rates of standard k-NN, while Relief and Simba often even deteriorate the classification performance. For the normalized datasets the results change drastically as it can be seen in Table 2. MDM is the only method for which there is almost no change compared to Table 1. LMNN and D-LMNN have a change in their results due to a different initial selection of target neighbors. Obviously, this selection depends on the intial scaling and can be improved

[1] A implementation is available at `www.inb.uni-luebeck.de/tools-demos/mdm/mdm.m`

Table 1. Results for unpreprocessed data. The top entry is the average test error followed by the variance in parentheses. Below the error rates the average rank, again followed by the variance, is given. In case of the feature weighting methods, the rank is equal to the non zero weights. The best results obtained with feature weighting are indicated by bold face.

	Euclidean	MDM	Relief	Simba	D-LMNN	LMNN
Iris	3.33(1.81)	**2.93(1.97)**	3.60(2.18)	4.40(1.89)	3.33(1.44)	3.07(1.89)
	4.00(0.00)	4.00(0.00)	4.00(0.00)	3.90(0.01)	4.00(0.00)	4.00(0.00)
Wine	31.00(4.52)	**4.00(2.23)**	34.67(3.05)	34.44(3.05)	4.33(1.99)	5.22(2.03)
	13.00(0.00)	11.50(0.52)	13.00(0.00)	12.90(0.01)	13.00(0.00)	12.30(0.21)
Breast	39.68(2.21)	**3.60(0.69)**	36.05(11.54)	39.77(2.21)	4.44(0.99)	3.80(0.57)
Cancer	10.00(0.00)	9.50(0.25)	9.90(0.01)	9.90(0.01)	10.00(0.00)	9.00(0.00)
Pima	30.78(1.99)	**28.49(1.87)**	30.08(2.19)	30.86(1.68)	29.14(2.62)	29.14(2.33)
Diabetes	8.00(0.00)	8.00(0.00)	7.50(0.08)	7.50(0.52)	8.00(0.00)	8.00(0.00)

Table 2. Results for the preprocessed datasets. The data dimensions of each dataset were normalized so that the data points have zero mean and a variance of one. The notation and structure of this table is the same as in Table 1.

	Euclidean	MDM	Relief	Simba	D-LMNN	LMNN
Iris	3.60(1.41)	2.93(1.97)	**2.53(1.33)**	3.73(1.64)	4.27(1.38)	2.67(1.99)
	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)
Wine	5.56(1.48)	**4.00(2.23)**	4.78(2.10)	4.56(2.06)	4.89(1.50)	2.67(1.90)
	13.00(0.00)	12.70(0.05)	13.00(0.00)	13.00(0.00)	13.00(0.00)	13.00(0.00)
Breast	3.92(0.63)	3.54(0.67)	3.65(0.57)	4.80(1.29)	**3.30(0.63)**	3.57(0.49)
Cancer	10.00(0.00)	10.00(0.00)	10.00(0.00)	9.70(0.05)	10.00(0.00)	9.70(0.05)
Pima	**27.76(1.44)**	28.49(1.87)	28.20(1.91)	29.35(2.97)	28.49(2.08)	27.92(1.71)
Diabetes	8.00(0.00)	8.00(0.00)	7.60(0.07)	6.60(3.32)	8.00(0.00)	8.00(0.00)

by an appropriate preprocessing. Especially LMNN relies on it. Relief and Simba work now as they are supposed to and become competitive. Even the standard Euclidean distance is becoming a quite good choice. Nevertheless, for all datasets except for the Pima Diabetes dataset the feature weighting and the metric learning methods perform best. MDM is still very competitive, but not clearly better than the others anymore. So the main advantage is that MDM is independent of the initial data scaling. Interestingly, LMNN performed significantly better than the feature weighting methods only on the Wine dataset. This shows that often the extra flexibility of learning the entire Mahalanobis distance is not needed.

4 Conclusion

We have presented a simple linear programming based method for feature weighting. By reweighting the dimensions of the dataset, the scaling is changed so that the classification performance of k-NN is improved significantly compared to using the standard Euclidean distance. However, if the dimensions of the datasets are normalized according to their variance prior to testing, the performance improvement drops since the other methods profit from this normalization. The experiments on the UCI datasets show that feature weighting is for most datasets competitive to the metric learning algorithm LMNN. This shows that the flexibility of learning the entire Mahalanobis distance is often not needed. Therefore, for very high dimensional datasets with few training points it is advisable to use the simpler feature weighting methods.

The algorithm we presented uses hard boundaries. It does not apply softness by using slack variables. This keeps the number of parameters to optimize low. The missing softness had no negative effect on the datasets we tested, but very noisy datasets with outliers might need some softness to yield robust results. Also, it would be interesting to see the effect of using target neighborhoods in MDM, which would adapt MDM more closely to the k-NN algorithm.

References

1. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
2. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
3. Gilad-Bachrach, R., Navot, A., Tishby, N.: Margin based feature selection - theory and algorithms. In: Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004, pp. 43–50. ACM, New York (2004)
4. Hammer, B., Villmann, T.: Generalized relevance learning vector quantization. Neural Netw. 15(8-9), 1059–1068 (2002)
5. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Proc. 9th International Workshop on Machine Learning, pp. 249–256 (1992)
6. Li, Y., Lu, B.L.: Feature selection based on loss-margin of nearest neighbor classification. Pattern Recogn. 42(9), 1914–1921 (2009)
7. Sun, Y., Li, J.: Iterative relief for feature weighting. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 913–920. ACM, New York (2006)
8. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems 19. MIT Press, Cambridge (2006)
9. Weinberger, K.Q., Saul, L.K.: Fast solvers and efficient implementations for distance metric learning. In: Proceedings of the 25th International Conference on Machine Learning, ICML 2008, pp. 1160–1167. ACM, New York (2008)
10. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. 10, 207–244 (2009)
11. Weinberger, K., Sha, F., Saul, L.: Convex optimizations for distance metric learning and pattern classification. Signal Processing Magazine 27, 146–158 (2010)

Training Computationally Efficient Smartphone–Based Human Activity Recognition Models[*]

Davide Anguita[1], Alessandro Ghio[1], Luca Oneto[1], Xavier Parra[2], and Jorge Luis Reyes-Ortiz[1,2]

[1] DITEN – University of Genoa, Via Opera Pia 11A, Genova, I-16145, Italy
{Davide.Anguita,Alessandro.Ghio,Luca.Oneto}@unige.it

[2] CETpD - Universitat Politècnica de Catalunya, Vilanova i la Geltrú 08800, Spain
{xavier.parra@upc.edu,jorge.luis.reyes@estudiant.upc.edu}

Abstract. The exploitation of smartphones for Human Activity Recognition (HAR) has been an active research area in which the development of fast and efficient Machine Learning approaches is crucial for preserving battery life and reducing computational requirements. In this work, we present a HAR system which incorporates smartphone-embedded inertial sensors and uses Support Vector Machines (SVM) for the classification of Activities of Daily Living (ADL). By exploiting a publicly available benchmark HAR dataset, we show the benefits of adding smartphones gyroscope signals into the recognition system against the traditional accelerometer-based approach, and explore two feature selection mechanisms for allowing a radically faster recognition: the utilization of exclusively time domain features and the adaptation of the L1 SVM model which performs comparably to non-linear approaches while neglecting a large number of non-informative features.

Keywords: Smartphones, Human Activity Recognition, SVM, L1 SVM, Feature Selection.

1 Introduction

Human Activity Recognition using wearable systems is attracting a growing interest due to the recent advances in technology, which are making available a wide range of microelectromechanical sensors and portable computing devices. It has mainly addressed healthcare-related applications such as the development of automated assisted living and ambulatory monitoring technologies for people with disabilities, going through rehabilitation, or the elderly, partially substituting the continuous supervision needed from caretakers or other family members [25,16]. HAR systems facilitate to better understand different aspects of the individuals daily living to provide context information suitable for

[*] This work was supported in part by the Erasmus Mundus Joint Doctorate in Interactive and Cognitive Environments, which is funded by the EACEA Agency of the European Commission under EMJD ICE FPA n 2010-0012.

© Springer-Verlag Berlin Heidelberg 2013

a range of applications and services. This is commonly done through the use of body sensors that measure various attributes derived from motion, location, physiological signals and the environment (e.g. accelerometers, GPS, hear rate monitors, microphones) and then by interpreting this sensory input data with Machine Learning approaches such as Artificial Neural Networks (ANN), SVMs, k-Nearest Neighbors (KNN) and the Decision Tree algorithm C4.5 [15,17].

Some recent HAR approaches have taken advantage of smartphones due to their powerful processing capabilities and opportunistic sensing [23] from the available embedded components. These smart devices have been massively produced integrating small scale and low cost inertial sensors aiming to enhance human-computer interaction (e.g. for gaming and user interfaces). Research on HAR exploiting smartphone motion sensors has mostly incorporated only accelerometers [12,15], as they have been earlier introduced in 2007 [11]. Gyroscopes have been more recently included in 2010, though some less recent approaches have considered special purpose devices for HAR purposes (e.g. [2,13]). Recently, in [29] a hybrid accelerometer and gyroscope approach was used for the classification of 9 activities using an iPhone 4. They showed insights of the benefits of adding gyroscope signals into the recognition system achieving improvements ranging from 3.1% to 13.4% in classification accuracy, though a limited set of features from the gyroscope signals (signal mean, standard deviation and sum of signal magnitude in a sliding window) was used. A major limitation for the smartphone implementation was due to the exploitation of a KNN classifier, which could become inadequate in such applications due to its demanding computations for prediction, particularly with large training sets.

Open rooms for improving smartphone-based HAR consequently exist: (i) the benefits of introducing a larger set of gyroscope-based signals should be more thoroughly evaluated; on the other hand, (ii) a proper selection of the most useful features and simple (though effective) models should be carried out, so to make HAR more suitable for devices with limited battery life and computational restrictions. This paper targets both these two issues. Regarding point (i), we exploit the HAR dataset [3], which contains gyroscope measures plus a large set of previously suggested features in the time and frequency domain [5,14,1,9,8], for our analyses. Concerning issue (ii), we resort to effective Support Vector Machine (SVM) classifiers [28] and we implement two feature selection mechanisms to allow faster and computationally non-intensive recognition: on the one hand, conversely to [18], we do not discriminate on proposed feature sets, but instead on sensor type and domain; on the other hand, L1 SVM models [27] will be implemented, allowing to perform an automatic selection of significant features emerging from the training set while keeping the appealing classification performance of conventional SVMs. However, one of the main drawbacks of L1 SVMs consists in the impossibility of resorting to non-linearity to improve classification accuracy through the kernel trick [28]: for targeting this issue, we compare the use of conventional non-linear models with the one of linear classifiers in the particular case of HAR, showing how the latter ones are characterized by better performance/complexity ratios than the former ones.

2 HAR Dataset

The Human Activity Recognition Dataset [3], that will be used in the forthcoming analyses and is available at [6], was developed by the authors for the experimentation with smartphone inertial sensors and the classification of Activities of Daily Living (ADL): *standing, sitting, laying, walking, walking upstairs* and *walking downstairs*, that we respectively define A1-A6 for the sake of simplicity.

To create the dataset, a group of 30 volunteers followed a protocol of activities while carrying an Android OS smartphone attached to a belt on their waist. The dataset includes 10299 patterns which have been divided in training and test sets in a proportion of 70% to 30%. Each pattern is represented with a feature vector of 561 elements composed of time and the frequency domain features extracted from the accelerometer and gyroscope signals. These were first preprocessed for noise reduction and the removal of the gravitational component from the acceleration signal. Then fixed-width sliding windows of 2.56 sec width and 50% overlap between them were extracted from the inertial signals. From each window, a vector of features was calculated including features such as mean, standard deviation, signal magnitude area, interquartile range, auto-regresion coefficients, largest Fast Fourier Transform (FFT) power spectrum component and correlation coefficients between signal pairs, etc. For further details, the reader can also refer to [3].

3 An Effective Smartphone-Based Solution for HAR

Our target is to design a model, which can be effectively run on smartphones with limited battery life and computational restrictions. We have thus to identify the simplest possible classifier exploiting the smallest set of features that guarantees the best performance/computational burden ratio. For these purposes, we peruse the exploitation of linear models, which use only those selected inputs that are crucial to attain sufficient classification accuracy. The following subsections are devoted to describing the analyses, that have been performed on the previously introduced dataset.

3.1 Non-linear vs. Linear SVMs

SVM is one of the most widely employed Machine Learning algorithms [28,26,7]. It allows to solve binary problems by deterministically finding a maximum-margin hyperplane that separates the data. The effectiveness of SVM models make them particularly appealing in several and heterogeneous real-world problems, including applications on smartphones [4].

Table 1. Confusion Matrix for the comparison of performance of linear and Gaussian L2 SVM

	Linear SVM							**Gaussian SVM**						
	A1	A2	A3	A4	A5	A6	%	A1	A2	A3	A4	A5	A6	%
A1	**492**	1	3	0	0	0	*99.2*	**492**	1	3	0	0	0	*99.2*
A2	18	**451**	2	0	0	0	*95.7*	18	**451**	2	0	0	0	*95.7*
A3	4	6	**410**	0	0	0	*97.6*	4	6	**410**	0	0	0	*97.6*
A4	0	2	0	**434**	55	0	*88.4*	0	2	0	**432**	57	0	*88.0*
A5	0	0	0	14	**518**	0	*97.4*	0	0	0	14	**518**	0	*97.4*
A6	0	0	0	0	0	**537**	*100.0*	0	0	0	0	0	**537**	*100.0*
%	*95.7*	*98.0*	*98.8*	*96.9*	*90.4*	*100.0*	***96.4***	*95.7*	*98.0*	*98.8*	*96.9*	*90.1*	*100.0*	***96.4***

Given a dataset $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_l, y_l)\}$, $\boldsymbol{x}_i \in \mathbb{R}^m$, and $y_i \in \{\pm 1\}$, an SVM classifier $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ is found by solving the following primal problem, which exploits an L2 regularization term to adjust the size of the class of functions [28]:

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \mathbf{1}_l^T \boldsymbol{\xi} \\ \text{s.t.} \quad & Y(X\boldsymbol{w} + \boldsymbol{b}_l) \geq \mathbf{1}_l - \boldsymbol{\xi}, \ \boldsymbol{\xi} \geq \mathbf{0}_l, \end{aligned} \tag{1}$$

where $\xi_i = \max[0, (1 - y_i f(\boldsymbol{x}_i))]$, $X = [\boldsymbol{x}_1 | \dots | \boldsymbol{x}_l]^T$, $\boldsymbol{y} = [y_1 | \dots | y_l]^T$, $Y = \text{diag}(\boldsymbol{y})$ (Y is a diagonal matrix where the element on the diagonal are the $y_{i \in \{1,\dots,n\}}$), and $\boldsymbol{a}_p$ is a vector of p elements all equal to a. By introducing n Lagrange multipliers $\boldsymbol{\alpha}$ we obtain the dual formulation of the conventional SVM:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{1}_l^T \boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{0}_l \leq \boldsymbol{\alpha} \leq \boldsymbol{C}_l, \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \end{aligned} \tag{2}$$

where $Q \in \mathbb{R}^{l \times l}$ and $Q_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $K(\cdot, \cdot)$ is the kernel function. As we are targeting multiclass classification problems, generalization through the One-Vs.-All (OVA) approach is exploited [22,20].

The first performed experiments aim at comparing the performance of SVM models, based on linear and Gaussian kernels. The confusion matrices in Table 1 depict the classification results given the 6 ADL using the complete set of features introduced in Section 2. The accuracies achieved with the two methods are substantially identical, thus showing the equivalence between these two models. The linear approach is consequently preferred for the prediction of activities, more specifically for its application in limited resources devices: in fact, the prediction phase is much faster than the kernelized approach and linear models allow to exploit more sophisticated dimensionality reduction approaches, as will be shown in the next sections. Results also show balanced precision and recall measures for all the activities.

Table 2. Experiments with different feature subsets and conventional linear SVM models

Acc	Gyro	Time	Freq	Feature groups	N. Features	Linear SVM
0	1	1	0	GT	108	78.0%
0	1	1	1	GTF	213	81.0%
1	0	1	0	AT	164	90.6%
1	0	1	1	ATF	348	91.2%
1	1	1	0	AGT	272	95.8%
1	1	1	1	AGTF	561	96.4%

3.2 Selection of Subsets of Features

The second set of analyses consists in evaluating the inputs available in the dataset aiming at a reduction in the number of significant features. This is achieved by separating the inputs in groups with respect to: the type of sensor employed, namely *Accelerometer* (*A*) and *Gyroscope* (*G*); the domain, namely *Time* (*T*) and *Frequency* (*F*).

In practice, we expect to balance the trade-off between the addition of meaningful features and the removal of the ones that are redundant or that require expensive computations for their estimation. For such purposes, we test all the combinations of feature groups and compute the test error rate performed by a linear SVM model, trained on the subset employed: Table 2 presents the results. They suggest that the whole set of features (AGTF) should be exploited, although frequency-related inputs are not strictly necessary for this application as they do not largely affect recognition performance while, contrarily, requiring a remarkable computational effort for their derivation. Results also allow to gather some evidence of the benefits, which the addition of gyroscope signals bring into the HAR system, thus counterbalancing the (limited) slowdown in prediction due to the presence of these extra features. Note however that the models trained with sets using only gyroscope features (GT, GTF) have a lower performance, suggesting that the use of this sensor by its own is not appropriate for this application, despite enhancing the recognition when exploited concurrently with accelerometers.

3.3 Dimensionality Reduction with L1-SVM

As the conventional SVM does not perform any dimensionality reduction [10], which is desirable in some practical applications to highlight relevant features as well as to reduce the computational burden of performing the classification of new samples, the replacement of the L2 term with an L1 Manhattan norm based regularization has been proposed (L1 SVM) [27]:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \|\boldsymbol{w}\|_1 + C\mathbf{1}_l^T\boldsymbol{\xi}, \\ \text{s.t.} \quad & Y\left(X\boldsymbol{w}+\boldsymbol{b}_l\right) \geq \mathbf{1}_l - \boldsymbol{\xi},\ \boldsymbol{\xi} \geq \mathbf{0}_l. \end{aligned} \tag{3}$$

Table 3. Comparison of results using L2 and L1 SVM models

Feature groups	N. Features	L2 (linear) SVM	L1 SVM	N. Features L1 SVM	ρ
AGT	272	95.8%	96.5%	174	64.0 %
AGTF	561	96.4%	96.5%	275	49.0 %

Note that the conventional kernel trick cannot be exploited in the previous formulation, thus the effectiveness of linear models assumes an ever greater importance. This problem is a standard Linear Programming problem, for which many tools have been developed throughout the years [19].

L1 SVM allows to perform dimensionality reduction thanks to the characteristics of the Manhattan norm exploited, that is several weights w_i will be (generally) nullified during the learning procedure: this is in contrast with the conventional (L2) SVM, where $w_i \neq 0 \; \forall i = 1, ..., m$ in the final model. As we are exploiting OVA for targeting multiclass classification, we will consider a feature as filtered if the corresponding weight is null for all the (six) models learned for the OVA approach.

Table 3 presents the comparison of linear (L2) SVMs and L1 SVMs, both in terms of accuracy and number of selected features (remembering that L2 procedures do not perform any dimensionality reduction). In particular, we considered only the groups of features that showed to be necessary for HAR purposes, according to the results derived in the previous section. It is worth noting that L1 models perform comparably to (and, unexpectedly from literature, slightly better than) L2 models, furthermore allowing to remarkably reduce the dimensionality of the problem. The remarkable classification performance of L1 models is probably due to the filtering of noisy features, which negatively afflict L2 classifiers.

As a final remark, the results obtained with the L1 SVM algorithm also outperform the ones obtained at the ESANN 2013 HAR competition [21] in which contestants were challenged to propose novel approaches for the recognition of activities using the same HAR dataset. Linear and non linear Machine Learning methods were proposed, achieving a maximum classification accuracy of 96.4% with the work presented in [24], where an One-Vs.-One (OVO) SVM classification approach [22] was employed for the recognition task.

4 Conclusions

In this paper, we showed the benefits of adding gyroscope information into a human activity recognition system based on smartphone technology. We verify that a set of common daily activities can be accurately classified when this sensor is used along with the accelerometer. We explored three SVM algorithms including linear (L1 SVM, conventional L2 linear SVM) and non-linear (L2 Gaussian SVM) approaches on the HAR dataset and found a similar performance in terms of accuracy, but our selection criterion was subject to prediction speed and the possibility of applying them in devices with limited resources to provide less computational complexity and energy consumption.

Linear approaches exhibited the best trade off between accuracy and prediction speed, conferring distinctive benefits to the L1 SVM, which provides itself a reduction of the effective number of features needed for the prediction of the ADL. Furthermore, the study between different feature domains lead us to disregard frequency domain features as they were not only marginally contributing to the recognition performance but also adding expensive computations for their estimation. The ideal model selected for our application was the AGT, which only takes into account time domain features from the accelerometer and the gyroscope.

Future work will explore novel model selection approaches on the L1 SVM algorithm, that can help to further reduce the number of effective features by considering near-optimal hyperparameter models within the OVA binary classifiers to increase the number of zero-valued weights matches.

References

1. Allen, F.R., Ambikairajah, E., Lovell, N.H., Celler, B.G.: Classification of a known sequence of motions and postures from accelerometry data using adapted gaussian mixture models. Physiological Measurement 27(10), 901–935 (2006)
2. Altun, K., Barshan, B.: Human activity recognition using inertial/Magnetic sensor units. In: Salah, A.A., Gevers, T., Sebe, N., Vinciarelli, A. (eds.) HBU 2010. LNCS, vol. 6219, pp. 38–51. Springer, Heidelberg (2010)
3. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN (2013)
4. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: Bravo, J., Hervás, R., Rodríguez, M. (eds.) IWAAL 2012. LNCS, vol. 7657, pp. 216–223. Springer, Heidelberg (2012)
5. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
7. Ghio, A., Anguita, D., Oneto, L., Ridella, S., Schatten, C.: Nested sequential minimal optimization for support vector machines. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 156–163. Springer, Heidelberg (2012)
8. Karantonis, D.M., Narayanan, M.R., Mathie, M., Lovell, N.H., Celler, B.G.: Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. IEEE Transactions on Information Technology in Biomedicine 10(1), 156–167 (2006)
9. Khan, A.M., Lee, Y.K., Lee, S., Kim, T.S.: Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis. In: IEEE International Conference on Future Information Technology (2010)
10. Khan, N.M., Ksantini, R., Ahmad, I.S., Guan, L.: A sparse support vector machine classifier with nonparametric discriminants. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 330–338. Springer, Heidelberg (2012)

11. Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T.: A survey of mobile phone sensing. IEEE Communications Magazine 48(9), 140–150 (2010)
12. Lara, O., Labrador, M.: A survey on human activity recognition using wearable sensors. IEEE Communications Surveys Tutorials 1(99), 1–18 (2012)
13. Lee, S.W., Mase, K.: Activity and location recognition using wearable sensors. IEEE Pervasive Computing 1(3), 24–32 (2002)
14. Lovell, N., Wang, N., Ambikairajah, E., Celler, B.G.: Accelerometry based classification of walking patterns using time-frequency analysis. In: IEEE Annual International Conference of the Engineering in Medicine and Biology Society (2007)
15. Mannini, A., Sabatini, A.M.: Machine learning methods for classifying human physical activity from on-body accelerometers. Sensors 10(2), 1154–1175 (2010)
16. Narayanan, M.R., Scalzi, M.E., Redmond, S.J., Lord, S.R., Celler, B.G., Lovell, N.H.: A wearable triaxial accelerometry system for longitudinal assessment of falls risk. In: Annual International Conference of the IEEE Engineering in Medicine and Biology Society (2008)
17. Nishkam, R., Nikhil, D., Preetham, M., Littman, M.L.: Activity recognition from accelerometer data. In: Conference on Innovative Applications of Artificial Intelligence (2005)
18. Preece, S.J., Goulermas, J.Y., Kenney, L.P., Howard, D.: A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. IEEE Transactions on Biomedical Engineering 56(3), 871–879 (2009)
19. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes: The art of scientific computing, 3rd edn. Cambridge University Press (2007)
20. Ramírez, F., Allende, H.: Dual support vector domain description for imbalanced classification. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 710–717. Springer, Heidelberg (2012)
21. Reyes-Ortiz, J.L., Ghio, A., Anguita, D., Parra, X., Cabestany, J., Catal, A.: Human activity and motion disorder recognition: Towards smarter interactive cognitive environments. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN (2013)
22. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. The Journal of Machine Learning Research 5, 101–141 (2004)
23. Roggen, D., Förster, K., Calatroni, A., Holleczek, T., Fang, Y., Tröster, G., Lukowicz, P., Pirkl, G., Bannach, D., Kunze, K., Ferscha, A., Holzmann, C., Riener, A., Chavarriaga, R., del, R., Millán, J.: Opportunity: Towards opportunistic activity and context recognition systems. In: Proc. 3rd IEEE WoWMoM Workshop on Autononomic and Opportunistic Communications (2009)
24. Romera-Paredes, B., Aung, H., Bianchi-Berthouze, N.: A one-vs-one classifier ensemble with majority voting for activity recognition. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN (2013)
25. Ryder, J., Longstaff, B., Reddy, S., Estrin, D.: Ambulation: A tool for monitoring mobility patterns over time using mobile phones. In: IEEE International Conference on Computational Science and Engineering (2009)
26. Schölkopf, B., Smola, A.J.: Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press (2001)
27. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267–288 (1996)
28. Vapnik, V.: Statistical learning theory. Wiley-Interscience (1998)
29. Wu, W., Dasgupta, S., Ramirez, E.E., Peterson, C., Norman, G.J.: Classification accuracies of physical activities using smartphone motion sensors. Journal of Medical Internet Research 14(5), 105–130 (2012)

A Novel Procedure for Training L1-L2 Support Vector Machine Classifiers

Davide Anguita[1], Alessandro Ghio[1], Luca Oneto[1], Jorge Luis Reyes-Ortiz[1,2,⋆], and Sandro Ridella[1]

[1] DITEN – University of Genoa, via Opera Pia 11A, Genova, I-16145, Italy
{Davide.Anguita,Alessandro.Ghio,Luca.Oneto,Sandro.Ridella}@unige.it
[2] CETpD - Universitat Politècnica de Catalunya, Vilanova i la Geltrú 08800, Spain
jorge.luis.reyes@estudiant.upc.edu

Abstract. In this work we propose a novel algorithm for training L1-L2 Support Vector Machine (SVM) classifiers. L1-L2 SVMs allow to combine the effectiveness of L2 models and the feature selection characteristics of L1 solutions. The proposed training approach for L1-L2 SVM requires a minimal effort for its implementation, relying on the exploitation of well-known and widespread tools already developed for conventional L2 SVMs. Moreover, the proposed method is flexible, as it allows to train L1, L1-L2 and L2 SVMs, as well as to fine tune the trade-off between dimensionality reduction and classification accuracy. This scope is of clear importance in applications on resource-limited devices, such as smartphones, like the one we consider to verify the main advantages of the proposed approach: the UCI Human Activity Recognition real-world dataset.

Keywords: Support Vector Machine, Sequential Minimal Optimization algorithm, L1-L2 Regularization, Human Activity Recognition.

1 Introduction

The conventional L2-regularization Support Vector Machine (SVM) approach [15,11] is considered as one of the state-of-the-art methods for classification, and several effective techniques have been developed throughout the years for training L2 SVM models [9,10,5]. While allowing to derive sparse classifiers (i.e. models described by exploiting a limited subset of training patterns), L2 SVM [15] does not perform any feature selection nor reduction, representing a limitation for the analysis of the dataset and the interpretability of the informative content of the inputs. On the other hand, an alternative approach for overcoming this issue is the substitution of the L2 regularization term with the L1 [14], which allows to introduce in the learning process an automatic dimensionality reduction effect. However, despite being appealing, L1 SVMs are also characterized by some drawbacks: (i) no feature grouping effect characterizes L1 models,

⋆ This work was supported in part by the Erasmus Mundus Joint Doctorate in Interactive and Cognitive Environments, which is funded by the EACEA Agency of the European Commission under EMJD ICE FPA n 2010-0012.

© Springer-Verlag Berlin Heidelberg 2013

i.e. clusters of highly cross-correlated inputs are usually not entirely selected by the training procedure [12]; (ii) when the dimensionality of the dataset is remarkably larger than the number of samples, L1 models are able to exploit only a number of inputs at most equal to the cardinality of the training set, which could be restrictive in some applications [16]; (iii) L1 SVMs require custom ad hoc algorithms to be implemented for classifier training [8], which, despite being effective, do not allow to exploit the know-how stockpiled for the conventional L2 SVM learning problem in the last years (e.g. [9,10]).

In order to deal with points (i) and (ii) above, L1-L2 SVM has been proposed [16]. It allows to enhance feature grouping effects in model training, to properly balance sparsity and dimensionality reduction, and to combine the effectiveness of the L2 approach and the feature selection characteristics of L1 SVMs. In this paper we cope instead with issue (iii) and present a new training tool allowing to deal with L1, L2 and L1-L2 SVMs, without requiring that any special purpose custom optimization approach is developed. The proposal builds on the most widespread solvers designed in the last decades for L2 SVM (e.g. [10,9]), and thus can be implemented with a minimal effort. We test our algorithm on a Human Activity Recognition (HAR) dataset [2], publicly available at the well-known UCI repository [7] and conceived for the recognition of daily life activities of people carrying smartphones. In this particular application, reducing the number of features is important for understanding which are the most meaningful inputs, so to reduce energy consumption on mobile devices and to allow the recognition application running in background during the entire day.

2 L1-L2 SVM: Theory and Practice

In the framework of supervised learning, the goal is to approximate the relationship between examples from a set $\mathcal{X}$ and outputs from a set $\mathcal{Y}$: as we are targeting binary classification problems, we assume here that $\mathcal{X} \in \mathbb{R}^d$ and $\mathcal{Y} \in \{\pm 1\}$. The relationship between examples and outputs is encapsulated by a fixed, but unknown, probability measure $\mathcal{P}$. A training set $D_n = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ is sampled according to $\mathcal{P}$. The learning algorithm maps D_n to $f \in \mathcal{F}$ with $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$ (a linear separator in the original space) and the accuracy in representing the hidden relationship $\mathcal{P}$ is measured with reference to a loss function $\ell(f(\boldsymbol{x}), y)$.

In this framework, the hard loss function $\ell_H(f(\boldsymbol{x}), y) = [1 - y \operatorname{sign}(f(\boldsymbol{x}))]/2$ seems the most natural choice, as it counts the number of misclassifications, but unfortunately it is non–convex. For this reason the hinge loss function $\ell_\xi(f(\boldsymbol{x}), y) = [1 - y\ f(\boldsymbol{x})]_+$ is exploited instead [15]. Thus by introducing an L2 regularization term to adjust the size of the class, according to the Structural Risk Minimization (SRM) principle [15], we derive the primal formulation of the L2 SVM:

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \boldsymbol{1}_n^T \boldsymbol{\xi}, \quad \text{s.t. } Y\left(X\boldsymbol{w} + \boldsymbol{b}_n\right) \geq \boldsymbol{1}_n - \boldsymbol{\xi},\ \boldsymbol{\xi} \geq \boldsymbol{0}_n, \tag{1}$$

where $\xi_i = \ell_\xi(f(\boldsymbol{x}_i), y_i)$, $X = [\boldsymbol{x}_1 | \dots | \boldsymbol{x}_n]^T$, $\boldsymbol{y} = [y_1 | \dots | y_n]^T$, $Y = \operatorname{diag}(\boldsymbol{y})$ (Y is a diagonal matrix where the element on the diagonal are the $y_{i \in \{1,\dots,n\}}$), and $\boldsymbol{a}_p$

is a vector of p elements all equal to a. By introducing n Lagrange multipliers $\boldsymbol{\alpha}$ we obtain the dual formulation of L2 SVM:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^T Y X X^T Y \boldsymbol{\alpha} - \mathbf{1}_n^T \boldsymbol{\alpha}, \quad \text{s.t. } \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \ \mathbf{0}_n \leq \boldsymbol{\alpha} \leq C\mathbf{1}_n, \tag{2}$$

that is the conventional Convex Constrained Quadratic Problem (CCQP) of SVM training where $\boldsymbol{w} = \sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i$ and b is the Lagrange multiplier of the equality constraint. In order to solve Problem (2), many techniques have been proposed [13]: for example, the well-known and widely used Sequential Minimal Optimization (SMO) [10,9] consists in iteratively updating the two $\alpha_{i\in\{1,\dots,n\}}$ that mostly violate the Karush-Kuhn-Tucker (KKT) conditions until convergence is reached.

As L2 SVM does not perform any dimensionality reduction, which is desirable in some practical applications to highlight relevant features as well as to reduce the computational burden of performing the classification of new samples, the replacement of the L2 term with an L1 one has been proposed [14]:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \|\boldsymbol{w}\|_1 + C\mathbf{1}_n^T\boldsymbol{\xi}, \quad \text{s.t. } Y\left(X\boldsymbol{w} + \boldsymbol{b}_n\right) \geq \mathbf{1}_n - \boldsymbol{\xi}, \ \boldsymbol{\xi} \geq \mathbf{0}_n, \tag{3}$$

as L1 regularization introduces an automatic feature selection effect. Problem (3) is a standard Linear Programming (LP) problem:

$$\begin{aligned} \min_{\boldsymbol{w}^+,\boldsymbol{w}^-,b,\boldsymbol{\xi}} \quad & \mathbf{1}_d^T\left(\boldsymbol{w}^+ + \boldsymbol{w}^-\right) + C\mathbf{1}_n^T\boldsymbol{\xi} \\ \text{s.t.} \quad & Y\left[X\left(\boldsymbol{w}^+ - \boldsymbol{w}^-\right) + \boldsymbol{b}_n\right] \geq \mathbf{1}_n - \boldsymbol{\xi}, \ \boldsymbol{\xi} \geq \mathbf{0}_n, \ \boldsymbol{w}^+, \boldsymbol{w}^- \geq \mathbf{0}_d, \end{aligned} \tag{4}$$

but the effective tools developed for the conventional L2 SVM cannot be applied for solving it.

In order to combine the effectiveness of L2 solutions and the dimensionality reduction capabilities of L1 models, L1-L2 SVM has been proposed [16]:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\lambda\|\boldsymbol{w}\|_2^2 + (1-\lambda)\|\boldsymbol{w}\|_1 + C\mathbf{1}_n^T\boldsymbol{\xi} \\ \text{s.t.} \quad & Y\left(X\boldsymbol{w} + \boldsymbol{b}_n\right) \geq \mathbf{1}_n - \boldsymbol{\xi}, \ \boldsymbol{\xi} \geq \mathbf{0}_n, \end{aligned} \tag{5}$$

where $\lambda \in (0,1]$ is a constant that balances sparsity characteristics with feature selection ability. It is worth noting that, if $\lambda \to 0$, we derive the L1 SVM, while $\lambda = 1$ leads to the conventional L2 SVM. We can introduce an identity matrix I of size $d \times d$ and reformulate Problem (5) as:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\eta}^+,\boldsymbol{\eta}^-} \quad & \frac{1}{2}\lambda\boldsymbol{w}^T\boldsymbol{w} + (1-\lambda)\mathbf{1}_d^T\left(\boldsymbol{\eta}^+ + \boldsymbol{\eta}^-\right) + C\mathbf{1}_n^T\boldsymbol{\xi} \\ \text{s.t.} \quad & Y\left(X\boldsymbol{w} + \boldsymbol{b}_n\right) \geq \mathbf{1}_n - \boldsymbol{\xi}, \ \boldsymbol{\xi} \geq \mathbf{0}_n \\ & I\boldsymbol{w} = \boldsymbol{\eta}^+ - \boldsymbol{\eta}^-, \ \boldsymbol{\eta}^+, \boldsymbol{\eta}^- \geq \mathbf{0}_d. \end{aligned} \tag{6}$$

By introducing the Lagrange multipliers $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\boldsymbol{\beta} \in \mathbb{R}^d$, we obtain the dual formulation of Problem (6):

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix}^T \begin{bmatrix}[YXX^TY] & [YX] \\ [X^TY] & [I]\end{bmatrix} \begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix} - \begin{bmatrix}\sqrt{\lambda}\mathbf{1}_n\\ \mathbf{0}_d\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix} \tag{7}$$

$$\text{s.t.} \quad \begin{bmatrix} \sqrt{\lambda}\boldsymbol{y} \\ \mathbf{0}_d \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = 0, \; \begin{bmatrix} \mathbf{0}_n \\ -\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \leq \begin{bmatrix} \frac{C}{\sqrt{\lambda}}\mathbf{1}_n \\ \frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d \end{bmatrix},$$

where $\boldsymbol{w} = \frac{\sqrt{\lambda}}{\lambda}\left(X^T Y \boldsymbol{\alpha} + \boldsymbol{\beta}\right)$ and b is the Lagrange multiplier of the equality constraint (details are omitted due to space limits). Problem (5) is convex, so is its dual formulation [4]. Therefore we can split Problem (7) in two subproblems and solve them iteratively until the solution is reached [4]. In particular, we can fix $\boldsymbol{\beta}$ to some constant value $\hat{\boldsymbol{\beta}}$ (satisfying the constrains) and reformulate Problem (7) as follows:

$$\boldsymbol{\alpha}_t^*, b_t : \quad \arg\min_{\boldsymbol{\alpha}} \quad P1(\boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}) = \frac{1}{2}\boldsymbol{\alpha}^T Y X X^T Y \boldsymbol{\alpha} + \left(X^T Y \hat{\boldsymbol{\beta}} - \mathbf{1}_n^T\right)\boldsymbol{\alpha} \qquad (8)$$
$$\text{s.t.} \quad \sqrt{\lambda}\boldsymbol{y}^T\boldsymbol{\alpha} = 0, \; \mathbf{0}_n \leq \boldsymbol{\alpha} \leq \frac{C}{\sqrt{\lambda}}\mathbf{1}_n,$$

where b_t is derived analogously to the conventional L2 SVM approach. As a matter of fact, Problem (8) is a simple reformulation of the conventional dual formulation of L2 SVM, for solving which we can exploit any of the several approaches proposed in the last decades [13]. If, instead, we fix $\boldsymbol{\alpha}$ to some constant value $\hat{\boldsymbol{\alpha}}$ (satisfying the constrains), we can reformulate Problem (7) as follows:

$$\boldsymbol{\beta}_t^* : \quad \arg\min_{\boldsymbol{\beta}} \quad P2(\boldsymbol{\beta}, \hat{\boldsymbol{\alpha}}) = \frac{1}{2}\boldsymbol{\beta}^T I \boldsymbol{\beta} + \left(\hat{\boldsymbol{\alpha}}^T Y X\right)\boldsymbol{\beta} \qquad (9)$$
$$\text{s.t.} \quad -\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d \leq \boldsymbol{\beta} \leq \frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d.$$

Problem (9) has a closed form solution, as we have to identify the minimum of a paraboloid (characterized by an identity Hessian matrix) in a box:

$$\boldsymbol{\beta}_t^* = \max\left[-\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d, \min\left[\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d, \left(\boldsymbol{\alpha}_t^T Y X\right)^T\right]\right]. \qquad (10)$$

Consequently, a possible approach for solving Problem (7) is detailed in Algorithm 1: solutions of Problems (8) and (9) are iteratively found, where, for the former, any of the methods surveyed by [13] can be used. Though other optimization techniques could be exploited (future works will cope with this issue), Algorithm 2 focuses on the adoption of SMO [10,9] for solving Problem (8): in fact, SMO is one of the most general and widespread optimization procedure, and it also allows the extension to non-linear models through the exploitation of the representer theorem and the kernel trick [15]. Note that, since, at every optimization step, two α_i coefficients are optimized by SMO, in order to better balance the overall L1-L2 optimization procedure we can run $\frac{n}{2}$ iterations (chosen accordingly to [4]) of SMO and, then, update $\beta_{i\in\{1,\ldots,d\}}$.

The proposed approach allows to solve L1-L2 problems by simply exploiting optimization tools already developed for the conventional L2 SVM formulation. Moreover, by simply tuning the value of λ, it is also possible to exploit the proposed procedure for solving L1 ($\lambda \to 0$), L2 ($\lambda = 1$) and L1-L2 ($0 < \lambda < 1$) SVM training problems.

Algorithm 1. Algorithm for solving Problem (7)

```
Data: $D_n$, $\lambda$, $C$, $\varepsilon$ and numerical precision $\epsilon$
Result: $\boldsymbol{w}^*, b^*$
$t = 0$, $\boldsymbol{\alpha}_t = \mathbf{0}_n$, $\boldsymbol{\beta}_t = \mathbf{0}_d$;
repeat
    $\boldsymbol{\alpha}_{t+1}, b_{t+1} = \arg\min_{\boldsymbol{\alpha}} P1(\boldsymbol{\alpha}, \boldsymbol{\beta}_t)$;
    $\boldsymbol{\beta}_{t+1} = \max\left[-\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d, \min\left[\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d, \left(\boldsymbol{\alpha}_{t+1}^T Y X\right)^T\right]\right]$ ;
    t = t + 1;
until $\left[\left(\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t-1}\|_2^2 + \|\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\|_2^2\right) < \epsilon\right]$;
return $\boldsymbol{w}^* = \frac{\sqrt{\lambda}}{\lambda}\left(X^T Y \boldsymbol{\alpha}_t + \boldsymbol{\beta}_t\right)$, $b = b_t$
```

Algorithm 2. Extended SMO (EX-SMO) algorithm for solving Problem (7)

```
Data: $D_n$, $\lambda$, $C$, $\varepsilon$ and numerical precision $\epsilon$
Result: $\boldsymbol{w}^*, b^*$
$t = 0$, $\boldsymbol{\alpha}_t = \mathbf{0}_n$, $\boldsymbol{\beta}_t = \mathbf{0}_d$;
repeat
    $\boldsymbol{\alpha}_{t+1}, b_{t+1} = \arg\min_{\boldsymbol{\alpha}} P1(\boldsymbol{\alpha}, \boldsymbol{\beta}_t)$ by running $\frac{n}{2}$ iteration of the SMO algorithm [9] ;
    $\boldsymbol{\beta}_{t+1} = \max\left[-\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d, \min\left[\frac{(1-\lambda)}{\sqrt{\lambda}}\mathbf{1}_d, \left(\boldsymbol{\alpha}_{t+1}^T Y X\right)^T\right]\right]$ ;
    t = t + 1;
until $\left[\left(\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t-1}\|_2^2 + \|\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\|_2^2\right) < \epsilon\right]$;
return $\boldsymbol{w}^* = \frac{\sqrt{\lambda}}{\lambda}\left(X^T Y \boldsymbol{\alpha}_t + \boldsymbol{\beta}_t\right)$, $b = b_t$
```

3 Case Study: HAR on Smartphones

Our efforts are mainly targeted towards having a flexible approach, easy to implement, allowing to seamlessly train L1, L1-L2 and L2 SVMs by simply tuning a parameter, without requiring that ad hoc optimization approaches are realized. This is of particular importance for SVM benchmarking in the framework of Human Activity Recognition (HAR) using resource limited devices, such as smartphones: in these cases, the accuracy of the classification must marry the sparsity of the representation, so that computations are limited and battery charge is spared. Thus, in order to test our approach, we used the HAR dataset [2] downloadable from the UCI repository [7] and composed of 7352 training and 2947 test samples. Six activities of daily living must be recognized from values, gathered from a waist-mounted Samsung Galaxy S II smartphone: walking (A1), walking upstairs (A2), walking downstairs (A3), sitting (A4), standing (A5), laying down (A6). In order to derive the experimental results, we replicated the methodology adopted in [3], and we exploited a K–Fold Cross Validation [1] approach with $k = 10$ for model selection purposes, by searching for the SVM hyperparameter C in the range $[10^{-4}, 10^2]$ among 20 points equally spaced in a logarithmic scale. In order to compare L1, L2 and L1-L2 SVM solutions, we set $\lambda \in \{0.001, 0.005, 0.01, 0.1, 0.5, 1\}$.

Table 1 compares the training times for the EX-SMO described in Algorithm 2 against commonly used solvers for L1 (the Simplex Method for LP - SMLP [6]) and L2 SVM (the conventional SMO [9]). EX-SMO performs comparably

Table 1. Training times (in hours)

L1 SVM	L1-L2 SVM						L2 SVM
	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$	
SMLP	EX–SMO						SMO
2.54	2.37	1.97	1.47	1.39	1.32	1.14	1.13

Table 2. Confusion matrices as λ is varied

L1 SVM

	A1	A2	A3	A4	A5	A6	%
A1	**494**	1	1	0	0	0	99.6
A2	27	**443**	1	0	0	0	94.1
A3	2	5	**412**	1	0	0	98.1
A4	0	4	0	**437**	50	0	89.0
A5	1	0	0	11	**520**	0	97.7
A6	0	0	0	0	0	**537**	100.0
%	94.3	97.8	99.5	97.3	91.2	100.0	**96.5**

L1-L2 SVM $\lambda = 0.001$

	A1	A2	A3	A4	A5	A6	%
A1	**494**	1	1	0	0	0	99.6
A1	27	**443**	1	0	0	0	94.1
A3	2	5	**412**	1	0	0	98.1
A4	0	4	0	**437**	50	0	89.0
A5	1	0	0	11	**520**	0	97.7
A6	0	0	0	0	0	**537**	100.0
%	94.3	97.8	99.5	97.3	91.2	100.0	**96.5**

L1-L2 SVM $\lambda = 0.005$

	A1	A2	A3	A4	A5	A6	%
A1	**494**	1	1	0	0	0	99.6
A2	24	**446**	1	0	0	0	94.7
A3	2	5	**412**	1	0	0	98.1
A4	0	4	0	**439**	48	0	89.4
A5	1	0	0	11	**520**	0	97.7
A6	0	0	0	0	0	**537**	100.0
%	94.8	97.8	99.5	97.3	91.5	100.0	**96.6**

L1-L2 SVM $\lambda = 0.01$

	A1	A2	A3	A4	A5	A6	%
A1	**494**	1	1	0	0	0	99.6
A2	24	**446**	1	0	0	0	94.7
A3	2	4	**413**	1	0	0	98.3
A4	0	4	0	**442**	45	0	90.0
A5	1	0	0	10	**521**	0	97.9
A6	0	0	0	0	0	**537**	100.0
%	94.8	98.0	99.5	97.6	92.0	100.0	**96.8**

L1-L2 SVM $\lambda = 0.1$

	A1	A2	A3	A4	A5	A6	%
A1	**494**	1	1	0	0	0	99.6
A2	11	**460**	0	0	0	0	97.7
A3	2	4	**414**	0	0	0	98.6
A4	0	2	0	**453**	36	0	92.3
A5	0	0	0	7	**525**	0	98.7
A6	0	0	0	0	0	**537**	100.0
%	97.4	98.5	99.8	98.5	93.6	100.0	**97.8**

L1-L2 SVM $\lambda = 0.5$

	A1	A2	A3	A4	A5	A6	%
A1	**495**	1	0	0	0	0	99.8
A2	15	**455**	1	0	0	0	96.6
A3	1	4	**414**	1	0	0	98.6
A4	0	3	0	**460**	28	0	93.7
A5	0	0	0	5	**527**	0	99.1
A6	0	0	0	0	0	**537**	100.0
%	96.9	98.3	99.8	98.7	95.0	100.0	**98.0**

L1-L2 SVM $\lambda = 1$

	A1	A2	A3	A4	A5	A6	%
A1	**496**	0	0	0	0	0	100.0
A2	13	**458**	0	0	0	0	97.2
A3	1	1	**417**	1	0	0	99.3
A4	0	2	0	**461**	28	0	93.9
A5	1	0	0	7	**524**	0	98.5
A6	0	0	0	0	0	**537**	100.0
%	97.1	99.3	100.0	98.3	94.9	100.0	**98.2**

L2 SVM

	A1	A2	A3	A4	A5	A6	%
A1	**496**	0	0	0	0	0	100.0
A2	13	**458**	0	0	0	0	97.2
A3	1	1	**417**	1	0	0	99.3
A4	0	2	0	**461**	28	0	93.9
A5	1	0	0	7	**524**	0	98.5
A6	0	0	0	0	0	**537**	100.0
%	97.1	99.3	100.0	98.3	94.9	100.0	**98.2**

Table 3. Accuracy, feature selection and grouping ability for the approaches

Method	Algorithm	% Accuracy	% ρ	% σ
L1 SVM	SMLP	96.5	30.2	0.0
L1-L2 SVM $\lambda = 0.001$	EX–SMO	96.5	30.2	0.0
L1-L2 SVM $\lambda = 0.005$	EX–SMO	96.6	33.4	0.0
L1-L2 SVM $\lambda = 0.01$	EX–SMO	96.8	45.9	10.8
L1-L2 SVM $\lambda = 0.1$	EX–SMO	97.8	86.0	60.5
L1-L2 SVM $\lambda = 0.5$	EX–SMO	98.0	94.0	90.6
L1-L2 SVM $\lambda = 1$	EX–SMO	98.2	100.0	100.0
L2 SVM	SMO	98.2	100.0	100.0

to SMO on L2 problems and outstrips SMLP on training L1 SVMs, albeit EX-SMO effectiveness tends to decrease as $\lambda \to 0$: this is expected, as we are using a QP tool to solve an (almost) LP problem. However, dimensionality reduction considerations should not be neglected as well. As we exploit an OVA approach, the final classification is carried out by contemplating the output of six models $f^*_{i\in\{1,...,6\}}(\boldsymbol{x}) = \boldsymbol{w}_i^{*T}\boldsymbol{x} + b_i^*$: consequently, we can create a set $\mathcal{S}$ which includes the indexes of the features $j \in \mathcal{S} : \{w_{i,j} = 0\}\,,\ i = 1, ..., 6$. In other words, we can compute the fraction of selected features $\rho = \frac{d-|\mathcal{S}|}{d}$, where $|\mathcal{S}|$ indicates the cardinality of the set: broadly speaking, ρ will decrease (or increase) with λ.

Table 2 reports the confusion matrices for L1, L1-L2 and L2 SVMs, obtained on the HAR dataset: we do not present results for different solvers as no differences are shown in them. As expected, the accuracy tends to increase with λ, that is L2 solutions are more effective than L1 ones. Though dimensionality reduction capabilities are maximized for L1 SVMs, feature grouping effects, namely the ability of the algorithm in selecting (or neglecting) clusters of highly cross-correlated inputs, are usually absent as $\lambda \to 0$, although they are desirable in order to have more insights on the informative content of each input [12]. In order to evaluate whether L1-L2 SVMs are able to overcome these L1-related issues, as expected from literature, we computed the correlation matrix $M^C \in \mathbb{R}^{d\times d}$ of X and we created feature clusters by joining the 10 most cross-correlated inputs. Our purpose was to verify the percentage σ of clusters features selected (or neglected) by the different procedures (ranging from L1 to L2 SVM): a high value for σ is obviously desirable. Results are shown in Table 3, where it is thus worth noting that: a very small subset of features (L1 SVM) is necessary to guarantee an acceptable classification performance, though grouping effects are limited; by balancing the effects of L1 and L2 regularization terms, accuracy is enhanced altogether with feature grouping; performance is maximized for L2 SVM, as also derived by the analysis of the confusion matrices. In the particular case of HAR using smartphones, as we are targeting the minimization of the computational burden to maximize battery duration and we are only partially interested in having insights on information content of each input, L1-L2 SVMs with (very) small values of λ are preferable, but this could not be, in general, the best choice. The advantage of a very flexible solver that copes with L1, L1-L2 and L2 SVMs, as the one presented in this paper, consists in the possibility of identifying the best application-dependent trade-off between performance and dimensionality reduction, at the expense of a very small implementation effort.

4 Conclusions

We proposed in this paper a novel approach for training L1-L2 SVM classifiers. The proposed method is characterized by two main advantages: (i) it is flexible, as it allows to solve L1, L1-L2 and L2 SVM problems and to properly tune the trade-off between dimensionality reduction and performance; (ii) it builds on conventional solvers, thus can be implemented with a minimal effort. Tests on a real-world problem, where sparsity must be carefully balanced with accuracy depending on application-dependent constraints, allowed to highlight the usefulness of such a flexible solver in practice. Further more exhaustive tests are undergoing in order to assess the performance of the approach in a wider set of problems settings, whose results could not be included in this paper due to space constraints.

Acknowledgment. The authors would like to thank the anonymous reviewers for their valuable and constructive comments and suggestions, which greatly contributed to improving the quality and readability of our work.

References

1. Anguita, D., Ghio, A., Oneto, L., Ridella, S.: In-sample and out-of-sample model selection and error estimation for support vector machines. IEEE Transactions on Neural Networks and Learning Systems 23(9), 1390–1406 (2012)
2. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: European Symposium on Artificial Neural Networks (2013)
3. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: Bravo, J., Hervás, R., Rodríguez, M. (eds.) IWAAL 2012. LNCS, vol. 7657, pp. 216–223. Springer, Heidelberg (2012)
4. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University (2004)
5. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research 9, 1871–1874 (2008)
6. Flannery, B.P., Press, W.H., Teukolsky, S.A., Vetterling, W.: Numerical recipes in c. Press Syndicate of the University of Cambridge, New York (1992)
7. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
8. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(1), 1 (2010)
9. 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)
10. Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines. Tech. Rep. msr-tr-98-14, Microsoft Research (1998)
11. Schölkopf, B., Smola, A.J.: Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press (2001)
12. Segal, M.R., Dahlquist, K.D., Conklin, B.R.: Regression approaches for microarray data analysis. Journal of Computational Biology 10(6), 961–980 (2003)
13. Shawe-Taylor, J., Sun, S.: A review of optimization methodologies in support vector machines. Neurocomputing 74(17), 3609–3618 (2011)
14. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267–288 (1996)
15. Vapnik, V.: Statistical learning theory. Wiley-Interscience (1998)
16. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67(2), 301–320 (2005)

Online Classification of Eye Tracking Data for Automated Analysis of Traffic Hazard Perception

Enkelejda Tafaj[1], Thomas C. Kübler[1], Gjergji Kasneci[2], Wolfgang Rosenstiel[1], and Martin Bogdan[3]

[1] Department of Computer Engineering, University of Tübingen, Germany
[2] Hasso-Plattner-Institute, Germany
[3] Department of Computer Engineering, University of Leipzig, Germany

Abstract. Complex and hazardous driving situations often arise with the delayed perception of traffic objects. To automatically detect whether such objects have been perceived by the driver, there is a need for techniques that can reliably recognize whether the driver's eyes have fixated or are pursuing the hazardous object (i.e., detecting fixations, saccades, and smooth pursuits from raw eye tracking data). This paper presents a system for analyzing the driver's visual behavior based on an adaptive online algorithm for detecting and distinguishing between fixation clusters, saccades, and smooth pursuits.

Keywords: classification, eye data, traffic hazard, perception.

1 Introduction

Driving is a complex task requiring proper visual functioning. According to Nagayama, more than 50% of collisions in road traffic occur due to missed or delayed hazard perception [10, 18]. An effective way to avoid such collisions would be to provide automated means for monitoring and analyzing the driver's visual behavior to detect entities that might have been overlooked.

According to the *scanpath theory* by Noton and Stark [11] a top-down internal cognitive model of what we see drives our eyes efficiently over a scene [13] involving six types of eye movements: fixations, saccades, smooth pursuits, optokinetic reflex, vestibulo-ocular reflex, and vergence [8]. As fixations, saccades, and smooth pursuits are the most relevant while driving, we will focus on these three types of eye movements. During a fixation the eye is kept relatively stable on an area of interest (AOI), whereas saccades correspond to rapid eye movements enabling the retinal part of sharpest vision (fovea) to fixate different areas of the scene [13]. Smooth pursuits occur whenever the eye follows a moving target [3].

The continuous analysis of the driver's visual behavior (i.e., in an online fashion) and the reliable detection and distinction between fixation clusters, saccades, and smooth pursuits is a crucial step towards the automated recognition

© Springer-Verlag Berlin Heidelberg 2013

of traffic hazards. Most prior research on the detection of fixations and saccades has primarily focused on the offline detection of these types of eye movements. Various approaches such as position, velocity or acceleration based algorithms, methods based on minimum spanning trees, hidden Markov models or Kalman filters have been proposed [1, 2, 4, 7, 9, 12, 14, 15, 17, 20]. Yet, these approaches show two main drawbacks: (i) they either require several clustering parameters as input, which makes them inadequate for online usage or (ii) they show poor performance in the detection of fixations and saccades in dynamic scenes. A further challenge arises when smooth pursuits have to be distinguished from saccades and fixations. Although new methods have been proposed [6, 19], their applicability to the detection of smooth pursuits in dynamic scenes is unclear.

In order to identify whether a hazard was perceived by the driver, the driver's visual scanpath has to be analyzed in real time while considering all entities that appear on the visual scene. If a relatively stable target was perceived, we would expect the driver's eye movements to be focused on that target, thus yielding a cluster of fixation points. Any algorithm for clustering such fixation points needs to work in an online fashion and be unparameterized with respect to the number of clusters, as new entities may appear on the scene. Note that the system has to know the driver's AOIs at any point in time. Furthermore, as the viewing behavior differs from person to person, an adaptive algorithm is needed.

In this work, we present a work-flow for the online analysis of hazard perception based on an adaptive online algorithm for the identification of and distinction between fixations, saccades and smooth pursuits.

2 Analysis of Driver's Visual Behavior

2.1 Detection of Fixations and Saccades

In [16], we presented an effective online clustering algorithm, that could distinguish between fixations and saccades by considering only the Euclidean distance between subsequent data points recorded by the eye tracker. The underlying model was based on the intuition that distances between subsequent fixation points will in general be shorter than distances between subsequent saccade points; that is, distances between subsequent fixation points would be generated from a specific Gaussian distribution and those between subsequent saccade points from another. This intuition gave rise to an online Bayesian mixture model. Figure 1 depicts the belief network for the model, the joint probability distribution of which is given by:

$$p(\mathbf{D}, \mathbf{z}|\Theta) = \prod_{i=1}^{T-1} p(z_i|\pi)p(d_i|\mu_{z_i}, \beta_{z_i})$$

where $\mathbf{z} = \{z_1, ..., z_{T-1}\}$, $z_i \in \{1, 2\}$, contains the indices of the mixture component chosen for distance d_i between subsequent points $(i, i+1)$, and Θ represents the model parameters with $\pi = \{\pi_1, \pi_2\} \sim Dir(\lambda)$ drawn from a symmetric

Dirichlet distribution and denoting the set of mixture parameters μ_{z_i} and β_{z_i}, i.e., the mean and precision of the Gaussian from which the distance d_i is drawn; more specifically, $d_i \sim N(\mu_{z_i}, \beta_{z_i})$. Note that both parameters are drawn from corresponding conjugate priors, i.e., $\mu_{z_i} \sim N(m, \tau)$ and $\beta_{z_i} \sim Gam(n, \gamma)$.

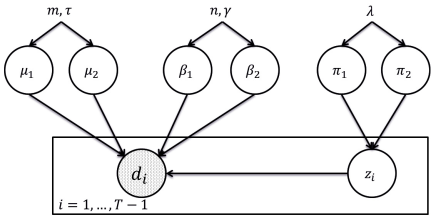


Fig. 1. Bayesian Mixture Model

The above Bayesian model comes with the great benefit that all parameters are updated and learned in an online fashion as more and more data is observed. More specifically, for every new data point, the priors of the parameters are substituted by the posteriors of the parameters. For practical purposes this means that for every new user the algorithm needs a relatively small number of data points to adjust to that user and learn user-dependent parameters.

For the specification of the model and inference in it we have used Infer.NET[1] and its variational message passing implementation.

2.2 Detection of Smooth Pursuits

Reliably distinguishing fixations from saccades is an important first step towards automated driving support and the prediction of hazardous situations. However, in driving scenarios, objects are typically in motion relative to the driver. Therefore it is crucial to automatically recognize objects that are being pursued by the driver's gaze and others that are not. To address this issue, we have extended the above model to recognize smooth pursuits.

We first describe the general idea and then the details of the algorithm.

Let us assume that the last k gaze points were labeled by the above mixture model as fixation points. The key question is whether these points are centered around a relatively stable target or correspond to a moving object that is being pursued by the user's gaze. In the former case, the vector that represents the highest variability in the k data points and the one representing the second highest variability will have approximately similar lengths. In the latter case there will be a notable difference in the lengths of the two vectors. Note that these vectors correspond to the first and the second eigenvectors of the covariance matrix of the data points. We rely on Principal Component Analysis [5] to efficiently retrieve these vectors.

[1] `http://research.microsoft.com/en-us/um/cambridge/projects/infernet/`

More specifically, let M be the matrix holding the last k subsequent data points that were all labeled as fixation points, such that the coordinates of the (empirical) mean of their distribution are subtracted from the coordinates of each point. Through singular value decomposition, M can be decomposed into $U\Sigma V^T$, where U contains the orthonormal eigenvectors of the covariance matrix MM^T, Σ is a diagonal matrix containing the positive roots of the eigenvalues of MM^T, and V contains the orthonormal eigenvectors of M^TM. This decomposition is unique up to different orderings of the values in Σ. This means that if the values are ordered decreasingly (with the largest value in the upper-left corner of the matrix) in Σ, then the first and the second column of U correspond to the first and second eigenvector of MM^T, respectively.

In order to decide whether the last k data points describe a smooth gaze pursuit, we compute

$$\frac{\sigma_2^2 \cdot \|u_2\|}{\sigma_1^2 \cdot \|u_1\|} = \frac{\sigma_2^2}{\sigma_1^2} \leq t$$

where t is an empirically established threshold, σ_1 and σ_2 are the largest and the second largest values in Σ, respectively, and the u_i the corresponding eigenvectors.

In preliminary experiments with data collected from driving simulations, this extension of the algorithm performed reliably and gave promising results. An example is depicted in Figure 2, where the hazardous situation arises from the white car cutting into the lane from the right. Figure 2(a) shows the moment when the driver's attention is caught for the first time by the white car. The black arrow shows the shift of visual attention in the most recent time frame of 1000ms. Figure 2(b) depicts the situation 400ms later, when the driver has come closer to the white car. During this time the driver's gaze has pursued the relative movement of the white car. According to the variability of the gaze points, our algorithm has classified them as a smooth pursuit.

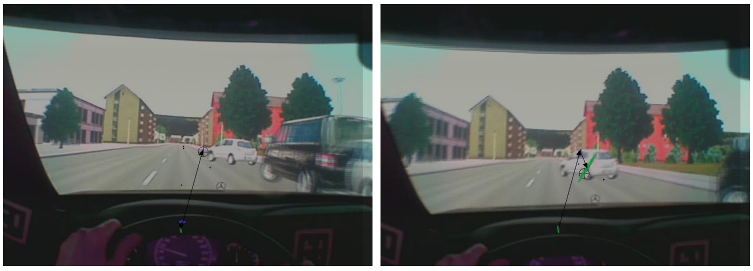

(a) (b)

Fig. 2. Smooth pursuit scenario

3 Experimental Results

The presented method for the online analysis of the driver's visual perception was evaluated in a simulator study with 27 subjects. The experiment was conducted in the moving-base driving simulator [21] shown in Figure 3(a) at the Mercedes-Benz Technology Center in Sindelfingen, Germany. The facility allows simulating acceleration forces in all directions, with up to $1g$ into the direction of a twelve-meter long rail. The cabin contained a real car body (Mercedes S class with automatic transmission) amidst a 360° projected virtual reality, Figure 3(b). The car body was oriented perpendicular to the rail. Thus curve and lane changing movements were simulated most realistically, while acceleration and braking resulted in a movement that was often described as "diving". All in all, acceleration, sound effects, and car environment contributed to a near-to-reality driving experience.

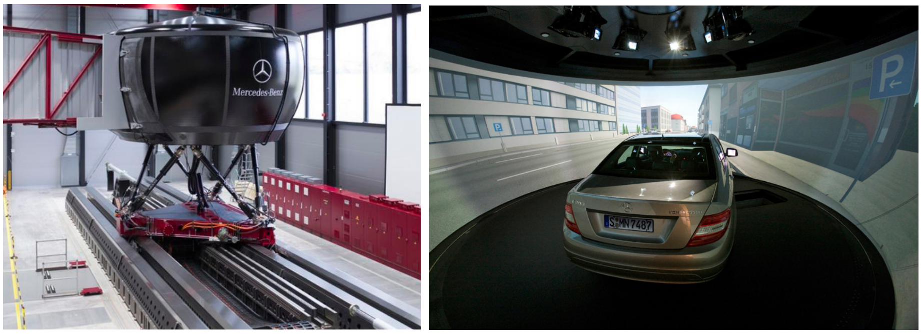


Fig. 3. Moving base driving simulator. The entire cabin is mounted on a hexapod, moving along the $12m$ rail resulting in up to $1g$ acceleration force[2].

The driving route of 37.5 km length contained ten hazardous situations. For 27 study subjects this would result in a total of 270 hazardous situations. However, some of the study participants aborted the session due to motion sickness, resulting in a total of 184 hazardous situations. The course contained rural as well as urban areas with different speed limits up to 100 km/h. Traffic hazards, e.g., pedestrians suddenly appearing behind parking cars and trying to cross the road (Figure 4) and risky overtaking maneuvers, were induced at various positions of the driving course. In case of an overseen hazard, the participants did not experience a crash. For example, it was not possible to run over pedestrians. Instead pedestrians would leap backwards and overtakers would return to their original lane to avoid crashes and subsequent psychological stress to the subjects.

Driver's eye movements were recorded at 25Hz using a Dikablis mobile eye tracker, whereas head movements were recorded by a LaserBird head tracker.

[2] Figures were provided by `http://www.daimler.com/`

Prior to the driving session each subject underwent a brief training session of 5 km length, in order to adjust to the car and the driving environment. This session served as an opportunity to learn an initial visual behavior model with parameters adjusted to the current driver. The training session began with a straight road and became more complex as oncoming traffic became successively denser and traffic signs more frequent. Complex urban driving scenarios were also part of this training session.

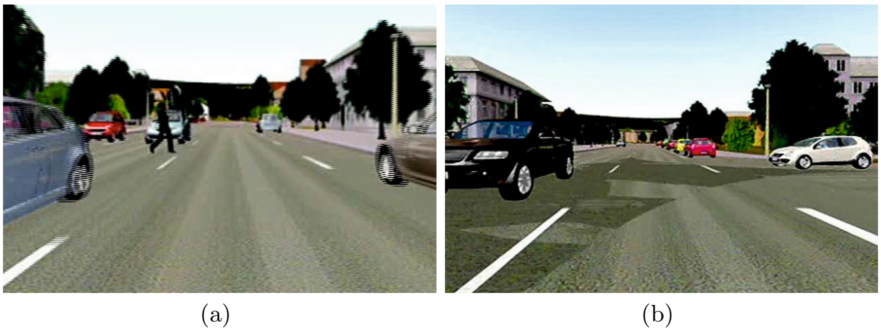


Fig. 4. Scenes from the virtual reality: (a) a pedestrian intending to cross the road, (b) and a white car coming from the right side

We evaluated our method on 184 hazardous situations. The spatial extent of the traffic hazards was manually annotated using bounding boxes. The analysis of the driver's viewing behavior and the detection of fixations, smooth pursuits and saccades was performed online using the algorithms presented in the previous section. A traffic hazard was considered as perceived, if a fixation or a smooth pursuit cluster intersected the bounding box around it. The results are presented in Table 1. In 169 situations, where the hazard was considered as perceived (i.e., an intersection between the fixation cluster and the bounding box occurred), the driver reacted by performing a braking or obstacle avoidance maneuver. An example of a successful intersection between the driver's gaze and the bounding box is shown in Figure 5. In 7 hazardous situations, the bounding box was not intersected by a fixation or smooth pursuit cluster and no reaction to the situation happened (i.e., the target was missed). Note that in a real-world scenario, these situations would have led to accidents. In 6 situations the bounding box was not intersected but the driver reacted nevertheless. These hazards were perceived, even though the driver did not explicitly look at them. In 2 situations, where the bounding box was intersected by a fixation or smooth pursuit cluster, the driver did not react. Although the driver looked at the targets, they were not perceived. Again, in reality, such situations would result in accidents. In terms of predicting the recognition of traffic hazards by the driver (using only raw eye-tracking data), the algorithm showed an overall accuracy of 95,7%, a specificity of 96,5%, and a sensitivity of 77,8%. These results highlight the overwhelming reliability of the proposed method.

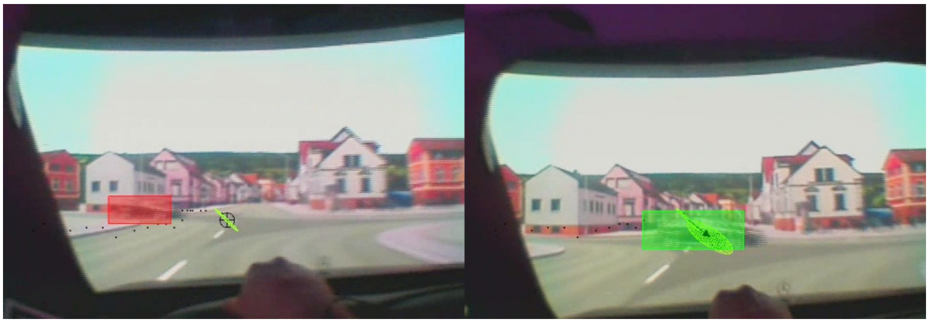


Fig. 5. A car appears on the left of the driving scene and is about to cut the driver's way. The red bounding box around the approaching hazard and the driver's fixation cluster are shown. Once the bounding box is intersected by the fixation cluster, the traffic hazard is marked as "perceived", highlighted by the green bounding box.

In order to look into the detailed per-class performance of our algorithm (i.e., the detection of fixations, saccades, and smooth pursuits), we randomly picked the raw eye tracking data of one of the subjects. For a six-minute-long driving sequence, two of our team members manually annotated the data points as fixations, saccades, or smooth pursuits. Note that this annotation task is very laborious, as the data has to be labeled frame-wise. Overall, there were 46 situations of the driver's visual behavior that were labeled; 27 of these as fixations, 8 as smooth pursuits, and 11 as saccades. Table 1 shows the per-class true-positive and false-positive counts. While our algorithm detected all saccades correctly, it classified two fixations as smooth pursuits and one smooth pursuit as fixation. It correctly identified 7 out of 8 smooth pursuits and 26 out of 27 fixations. Although preliminary in nature, these results are very promising and we plan to further evaluate the algorithm on larger labeled datasets.

Table 1. True and false positive counts for the detection of fixations, smooth pursuits, and saccades

Eye movement type	Annotation	TP	FP
Fixation clusters	27	26	1
Smooth pursuit clusters	8	7	2
Saccades	11	11	0

4 Conclusion

We presented an online adaptive, classification algorithm for detecting fixations, saccades, and smooth pursuits in driving scenarios. This algorithm was primarily evaluated with respect to its ability to detect hazardous traffic situations that might have been overlooked by the driver. In a user study with a state-of-the-art driving simulator, the method showed an impressive detection accuracy, which we think can be mainly explained by the method's ability to adjust the underlying model to the driver's visual behavior. A preliminary evaluation on the per-class detection of fixations, saccades, and smooth pursuits hints at the method's ability to recognize and distinguish between different types of eye movements. Apart from experiments on larger, labeled datasets, we also plan to investigate physiological models, which take heart rate and skin conductance into account to predict the driver's stress levels. Such models could supplement models that are based on gaze recordings to predict traffic hazard perception and the driver's ability to react.

Acknowledgment. The authors would like to thank Daimler AG and the driving simulator experts for the technical support during the whole experimental study. This research work was funded by DFG Grant RO 1030/15-1.

References

1. Berger, C., Winkels, M., Lischke, A., Höppner, J.: GazeAlyze: a MATLAB toolbox for the analysis of eye movement data. Behavior Research Methods 44(2), 404–419 (2012)
2. Camilli, M., Nacchia, R., Terenzi, M., Di Nocera, F.: Astef: A simple tool for examining fixations. Behavior Research Methods 40, 373–382 (2008)
3. Duchowski, A.: Eye tracking methodology: Theory and practice. Springer, London (2007)
4. Gitelman, D.R.: Ilab: a program for postexperimental eye movement analysis. Behavioral Research Methods, Instruments and Computers 34(4), 605–612 (2002)
5. Jolliffe, I.T.: Principal Component Analysis. Springer, New York (1986)
6. Komogortsev, O.V., Karpov, A.: Automated classification and scoring of smooth pursuit eye movements in the presence of fixations and saccades. Behavior Research Methods 45, 203–215 (2013)
7. Komogortsev, O.V., Gobert, D.V., Jayarathna, S., Koh, D., Gowda, S.: Standardization of automated analyses of oculomotor fixation and saccadic behaviors. IEEE Transactions on Biomedical Engineering 57, 2635–2645 (2010)
8. Leigh, R.J., Zee, D.S.: The neurology of eye movements. Oxford University Press (2006)
9. Munn, S.M., Stefano, L., Pelz, J.B.: Fixation-identification in dynamic scenes: comparing an automated algorithm to manual coding. In: Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization, APGV 2008, pp. 33–42. ACM, New York (2008)
10. Nagayama, Y.: Role of visual perception in driving. IATSS Research 2, 64–73 (1978)
11. Noton, D., Stark, L.W.: Eye movements and visual perception. Scientific American 224(6), 34–43 (1971)
12. Privitera, C.M., Stark, L.W.: Algorithms for defining visual regions-of-interest: Comparison with eye fixations. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(9), 970–982 (2000)
13. Privitera, C.M., Stark, L.W.: Scanpath theory, attention, and image processing algorithms for predicting human eye fixations. In: Itti, L., Rees, G., Tsotsos, J. (eds.) Neurobiology of Attention, pp. 269–299 (2005)

14. Salvucci, D., Goldberg, J.: Identifying fixations and saccades in eye-tracking protocols. In: Proceedings of the Eye Tracking Research and Applications, pp. 71–78 (2000)
15. Santella, A., DeCarlo, D.: Robust clustering of eye movement recordings for quantification of visual interest. In: Proceedings of the 2004 Symposium on Eye Tracking Research & Applications, pp. 27–34 (2004)
16. Tafaj, E., Kasneci, G., Rosenstiel, W., Bogdan, M.: Bayesian online clustering of eye movement data. In: Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA 2012, pp. 285–288. ACM, New York (2012)
17. Turano, K.A., Geruschat, D.R., Baker, F.H.: Oculomotor strategies for the direction of gaze tested with a real-world activity. Vision Research 43, 333–346 (2003)
18. Velichkovsky, B.M., Rothert, A., Kopf, M., Dornhöfer, S.M., Joos, M.: Towards an express-diagnostics for level of processing and hazard perception. Transportation Research Part F: Traffic Psychology and Behaviour 5(2), 145–156 (2002)
19. Vidal, M., Bulling, A., Gellersen, H.: Detection of smooth pursuits using eye movement shape features. In: Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA 2012, pp. 177–180. ACM, New York (2012)
20. Wooding, D.S.: Fixation maps: quantifying eye-movement traces. In: Proceedings of the Eye Tracking Research and Applications, pp. 31–36 (2002)
21. Zeeb, E.: Daimler's new full-scale, high-dynamic driving simulator-a technical overview. Actes INRETS, 157–165 (2010)

Time-Series Forecasting of Indoor Temperature Using Pre-trained Deep Neural Networks

Pablo Romeu, Francisco Zamora-Martínez,
Paloma Botella-Rocamora, and Juan Pardo

Embedded Systems and Artificial Intelligence Group
Escuela Superior de Enseñanzas Técnicas
Universidad CEU Cardenal Herrera
C/ San Bartolomé 46115 Alfara del Patriarca, Valencia, Spain
{pablo.romeu,francisco.zamora,paloma.botella,juan.pardo}@uch.ceu.es

Abstract Artificial neural networks have proved to be good at time-series forecasting problems, being widely studied at literature. Traditionally, shallow architectures were used due to convergence problems when dealing with deep models. Recent research findings enable deep architectures training, opening a new interesting research area called deep learning. This paper presents a study of deep learning techniques applied to time-series forecasting in a real indoor temperature forecasting task, studying performance due to different hyper-parameter configurations. When using deep models, better generalization performance at test set and an over-fitting reduction has been observed.

Keywords: Artificial neural networks, deep learning, time series, auto-encoders, temperature forecasting, energy efficiency.

1 Introduction

Time series forecasting is the task of predicting some future values of a given sequence, using historical data from the same signal (univariate forecasting), or using historical data from several correlated signals (multivariate forecasting). Artificial neural networks (ANNs) have been widely applied to this task [1,2,3]. Temperatures and energy consumption predictions could be applied to enhance energy efficiency in domotic environments [4]. Nowadays, deep ANNs achieve encouraging improvements over previous shallow ANN architectures.

Deep architectures, with many levels of non-linearity, have the ability of represent complex features from its inputs [5], yielding models with the theoretical ability to learn these complex features, achieving better generalization. With the exception of convolutional networks [6], deep ANN are difficult to train due to the non-convex training criterion used in the gradient descent algorithm [5,7,8]. This algorithm usually converges to different local minima, depending on the values of the parameter initialization. Empirical work has demonstrated [9] that training deep networks with more than 2 or 3 layers using random weights initialization and supervised training provide worse results than training shallow architectures.

© Springer-Verlag Berlin Heidelberg 2013

Due to the difficulty of training deep architectures in an efficient manner, deep networks have not been a trending topic in machine learning (also in time series forecasting) since the introduction of pre-training techniques [10]. The training is divided in two phases, unsupervised greedy layer-wise pre-training phase, and a supervised training phase.

Previous work on deep networks pre-training focuses on dimensionality reduction [10] or classification [11]. Time series forecasting with deep learning techniques becomes an interesting research area which needs to be studied. As far as we know, few works are available at literature which study time series forecasting using deep networks [12,13], principally following Restricted Boltzmann Machines (RBMs) approach proposed by [10]. This paper presents a study of Stacked Denoising Auto-Encoders (SDAEs) [11], another kind of deep architecture, and its ability to learn a real indoor temperature forecasting task [3]. An analysis of different hyper-parameters is shown, obtaining small but promising improvements using SDAEs. In the future it is expected that deep ANNs could learn trend and seasonality of time series improving the overall efficiency of the experimental framework.

2 Time Series Forecasting

Time series are data series with trend and pattern repetition through time. They might be formalized as a sequence of scalars of a given variable observations: $\bar{s} = s_0, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots$, denoting with s_i^j a fragment from time i to j.

Depending on the size of future window (H) and how it is computed [2], forecasting approaches are denoted as *single-step-ahead forecasting* (one future prediction); *multi-step-ahead iterative forecasting* (iterative future window); and *multi-step-ahead direct forecasting* [14] (large future window in one step). Following this last approach, *Multiple Input Multiple Output* (MIMO) variation uses one model to compute the full H future window. This paper follows MIMO using ANNs due to their ability to learn discriminatively the mapping between inputs and outputs. The forecasting model is formalized as a function F which receive as inputs the interest variable with its past values until current time t (input size I) and predicts a future window of size H following $\hat{s}_{t+1}^{t+H} = F(s_{t-I+1}^{t})$.

In this paper, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are used to compare the performance of the models. They are computed comparing target values for the time series $s_{t+1}, s_{t+2}, \ldots, s_{t+H}$ and its corresponding time-series prediction $\hat{s}_{t+1}, \hat{s}_{t+2}, \ldots, \hat{s}_{t+H}$:

$$\mathrm{MAE}(t) = \frac{1}{H} \sum_{h=1}^{H} |\hat{s}_{t+h} - s_{t+h}| \qquad \mathrm{RMSE}(t) = \sqrt{\frac{1}{H} \sum_{h=1}^{H} (\hat{s}_{t+h} - s_{t+h})^2} \, .$$

The results could be measured over all time series in a given dataset $\mathcal{D}$ as $\mathrm{MAE}^\star = \frac{1}{|\mathcal{D}|} \sum_{t=1}^{|\mathcal{D}|} MAE(t)$, being $|\mathcal{D}|$ the size of the dataset. In the same way is possible to compute $\mathrm{RMSE}^\star$.

3 Stacked Denoising Auto-Encoders

In order to solve deep architectures training problems, a greedy pre-training algorithm is used, which in practice works as a regularization mechanism, preparing the weights of the network so that the gradient descent reaches a better generalization result [8]

In recent approaches to this issue, Stacked Denoising Auto-Encoders (SDAEs) [9,11] have been proposed. By introducing small perturbations –i.e. noise– at the input data of the pre-training task, each layer of the deep network needs to recover a clean (denoised) input at the unsupervised phase. This approach tries to extract useful high-level features, and hence, be able to generalize better.

The SDAE training consists in two phases. At the first phase (*pre-training phase*) several auto-encoders (AEs) are trained, forcing each AE to reconstruct the encoding computed at the hidden layer of previous AE, except the first AE which is trained to reconstruct the input features of the task. SDAE [9,11] introduces noise to the input data of the pre-training task as shown in Figure 1. Therefore, the pre-trained layers extract robust features from data while trying to reconstruct its inputs. Once all layers have been pre-trained, there is a second supervised phase (*fine-tuning phase*) where the output layer is added and the whole ANN is trained to solve a concrete task.

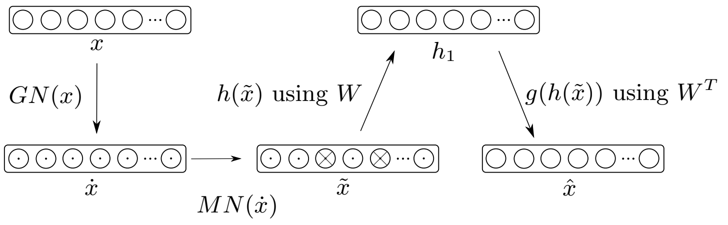


Fig. 1. Denoising Auto-Encoder with Gaussian (GN) and masking noisy (MN) inputs

Each AE computes its hidden layer activations, an *encoded* version of its inputs, as $h(x) = softsign(b + Wx)$ being x the input, b the bias, and W the weights of the AE. The activation function is $softsign(x) = \frac{x}{1+|x|}$ [15]. A transposed version of W weights matrix is used to *decode* hidden layer activations, following $g(h(x)) = softsign(c + W^T h(x))$, where c are the biases for the decoding layer. As the input for each layer is not the clean input x, but a corrupted version $\tilde{x}$, each pre-trained layer outputs $g(h(\tilde{x})) = softsign(c + W^T h(\tilde{x}))$. Then, this AE is trained in order to $g(h(\tilde{x}))$ minimizes the mean square error (MSE) between the clean input x and the output $g(h(\tilde{x})) = \hat{x}$. We define MSE of a m-sized mini-batch of patterns as: $MSE = \frac{1}{2m} \sum_{j=1}^{m} \sum_{i=1}^{n} (\hat{x}_i^{(j)} - x_i^{(j)})^2$, being $x_i^{(j)}$ the component i of the pattern j.

The types of noise added to the input data of each AE, following [11]), are:

– Gaussian noise (GN): $GN(x) = \dot{x} = x + \mathcal{N}(0, \sigma^2 I)$, where σ^2 is the variance of the amount of Gaussian noise added and I the identity matrix.
– Masking noise (MN): $MN(x) = \tilde{x}$, a percentage p of randomly chosen elements are forced to be 0 in case of inputs, and -1 for softsign activations.

4 Experimentation

Several experiments have been performed in order to study the effect of hyper-parameters, pre-training technique, and different fine-tuning schemes.

4.1 Experimentation Framework

Input signal sequence is sampled at one minute period (indoor temperature time series task presented at [3]), and pre-processed by a low-pass filter to get the mean value of the current plus last 14 samples, introducing a delay of 7 minutes in the predicted values. Hence, $s'_1 s'_2 \dots s'_N$ are computed, where $s'_i = (s_i + s_{i-1} + s_{i-2} + s_{i-3} + \dots + s_{i-14})/15$. In a second step, differences are calculated between each two adjacent elements, to estimate the variation of temperature within 15 minutes. Then $s''_1, s''_2 \dots s''_{N-1}$ are calculated, being $s''_i = s'_i - s'_{i+1}$.

Data is divided in three partitions: training (2 016 training patterns, 21 days), validation (672 validation patterns, 7 days) used during training to avoid over-fitting, and the last one for testing (672 test patterns, 7 days). The validation partition is sequential with the training partition, but the test partition is one week ahead from them.

Three types of training modes are compared among themselves. First mode is Train Mode 0 (TM-0) where an ANN is trained to forecast the data without pre-training. Train Mode 1 (TM-1) pre-trains an ANN using SDAE and fine-tuning all the layers. Train Mode 2 (TM-2) pre-trains the ANN using SDAE but fine-tuning only the last layer (forecasting layer).

Instead of using a grid search for hyper-parameter optimization, a combination of grid and random search is used. Compared to grid search, random search has demonstrated [16] to be more efficient in finding good hyper-parameter configurations within less trials and it is easier to parallelize. The Gaussian Noise variance and number of forecasted values were set to 0.01 and $H = 12$ respectively for all experiments. Mini-batch size was set to 32 in all cases. A grid with the following hyper-parameters was built: the Train Mode (TM-0, TM-1, TM-2); number of hidden layers (1, 2, 3); and Mask Noise percentage (0.02, 0.04, 0.10, 0.20). For each grid sweep 100 random trials have been performed sampling different values for the following hyper-parameters:

– *Input size* (I): uniform distribution (12, 24, 36, 48, 60, 72, 84, 96).
– *Learning rate*: two hyper-parameters, one for pre-train and other the fine-tuning, sampled uniformly and independently in range $[10^{-3}, 10^{-2}]$.
– *Momentum*: one hyper-parameter for each phase, independently sampled $\sim \mathcal{N}(10^{-3}, 5 \times 10^{-3})$, avoiding negative values.

– *Weight decay*: in TM-0, weight decay is used at the ANN training. In TM-1 and TM-2, weight decay is used at the pre-training phase, but not at fine-tuning. Uniformly sampled in the range $[0, 10^{-5}]$.
– *Hidden layer sizes*: uniformly distributed at range $[4, 1024]$, but taken only multipliers of 4 (to avoid memory alignment issues).

A minimum of 50 and maximum of 4 000 iterations over the training data are used. Training stops if ever, at iteration k, the best validation performance was observed before iteration $k/2$. In total 3 600 experiments were performed, all of them using the `April-ANN`[1] toolkit [17], which implements SDAE and efficient ANN training algorithms.

4.2 Results

The first noticeable result which we observe is that for TM-0, deeper networks were difficult to train. Approximately 58% of experiments for three layered ANNs, and the 33% for two layered ANNs, achieve MAE* greater than 0.5. This issue is not found for TM-1 and TM-2 experiments.

The Figure 2 shows validation MAE* results for different hyper-parameters. The input size behavior for each training mode (Figure 2-a), shows similar shape for TM-0 and TM-1, but worst results for TM-2. Best TM-0 and TM-1 results are between 48 and 60 input sizes (12–15 hours). This is coherent with the input signal observed frequency, where 12 hours past info seems to be enough to forecast the slope of the function, and therefore, to restrict next forecasted values to a short range. The lack of full fine-tuning harms the performance of TM-2 while adding hidden layers. Encoding layer (the number of neurons at the last hidden layer) results at Figure 2-b show consistently that rising the number of hidden layers introduces instability to the prediction results on TM-2. TM-0 and TM-1 remain more stable while increasing the number of hidden layers. Figure 2-c shows that masking noise harms the performance of TM-0. Comparing learning rates at fine-tuning phase, shown at Figure 2-d, we observe that TM-0 needed a higher learning rate to achieve good results, while for pre-trained TM-1 it is not an important parameter. All four figures show that as more hidden layers are added, the more unstable TM-2 is.

Test results in MAE* and RMSE* for the systems which perform better in validation are shown at Figure 3. The systems were optimized separately following the random search algorithm of [16]. The best system topologies, regarding to validation set performance, are: for TM-0 60 inputs and two hidden layers of 756 and 60; for TM-1 48 inputs and three hidden layers of 648, 920 and 16; and for TM-2 96 inputs and one hidden layer of 712. Non pre-trained ANNs TM-0 achieve similar errors as pre-trained ANNs TM-1. Nevertheless, performance measured at test error is better at full pre-trained ANNs TM-1. Pre-trained ANNs TM-2 achieve worse error measures than TM-0 and TM-1. In order to compare, an exponential smoothing model [18] was trained, denoted as *ETS* at the table of Figure 3.

[1] Developed by members of our research group in collaboration with members of Universitat Politècnica de València.

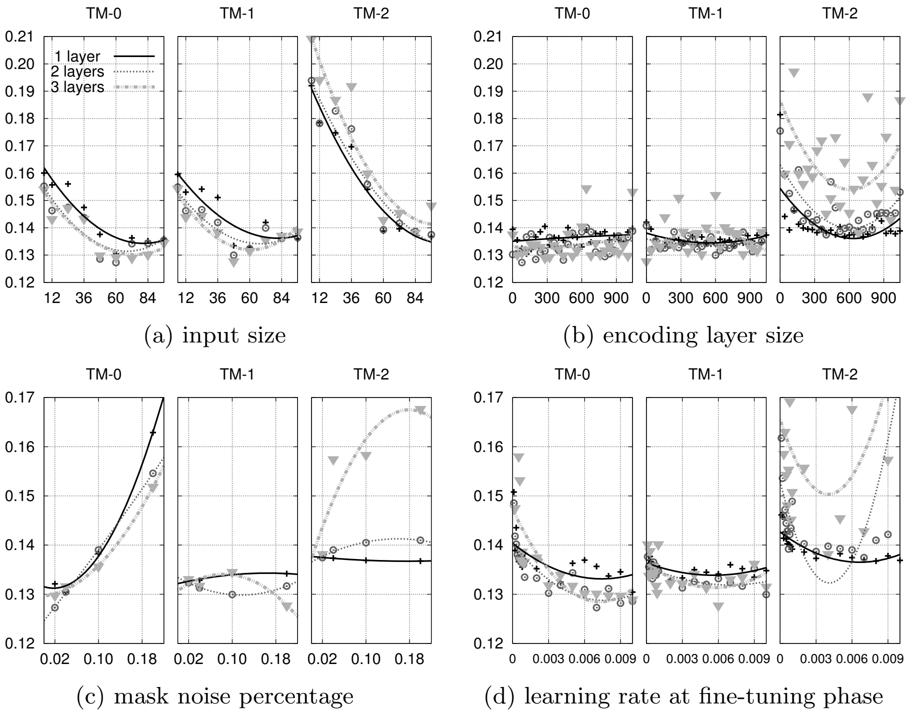
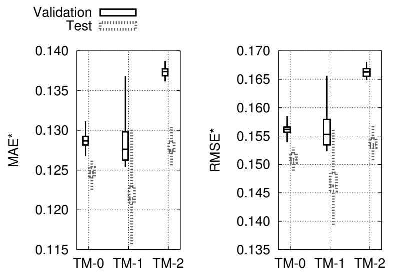


Fig. 2. Result plots of different hyper-parameters (x-axis) vs MAE* (y-axis). Only the best model for each x value is represented. Second order polynomial fits are also shown.



MAE*

	Validation ($\mu \pm \sigma$)	Test ($\mu \pm \sigma$)
ETS	0.3004	0.3254
TM-0	**0.1289 ± 0.0011**	0.12482 ± 0.0010
TM-1	**0.1287 ± 0.0033**	**0.1223 ± 0.0033**
TM-2	0.1374 ± 0.0007	0.1279 ± 0.0011

RMSE*

	Validation ($\mu \pm \sigma$)	Test ($\mu \pm \sigma$)
ETS	0.3648	0.3930
TM-0	**0.1563 ± 0.0011**	0.1511 ± 0.0012
TM-1	**0.1565 ± 0.0040**	**0.1473 ± 0.0039**
TM-2	0.1663 ± 0.0009	0.1538 ± 0.0013

Fig. 3. (Left) Box whiskers plot computed over 20 different random initializations for each training mode (TM) and for the validation and test sets. (Right) Mean and standard deviation for MAE* & RMSE* computed over the same 20 random initializations. Bolded numbers are best results. ETS is an exponential smoothing model.

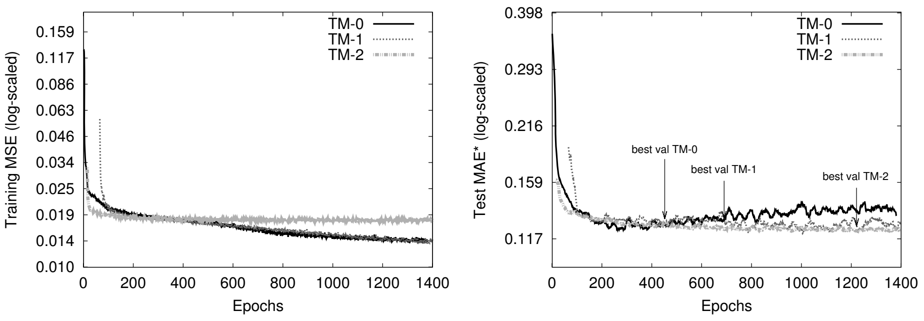


Fig. 4. Training detail for the best initialization (from the 20 random initializations tested for Figure 3). (Left) Plot of MSE at each training epoch for best ANN of each training mode. (Right) Plot of MAE* at each training epoch for best ANN of each training mode. Arrows indicates stopping point due to validation stopping criteria.

To conclude, learning curve and test set generalization of the best configurations are shown at Figure 4. Left plot shows that as training epochs increase, TM-2 stops learning at a certain epoch, while TM-0 and TM-1 keep improving. Right plot shows that TM-0 ANN over-fits if it is trained during too much epochs, while pre-trained networks remain close to its minimum error, showing the benefits of pre-training as a regularization method.

5 Discussion and Conclusions

A study of deep learning applied to time-series forecasting has been presented. Pre-training, denoising techniques, and random hyper-parameter optimization were used to carry deep ANNs training, showing better generalization performance at test set and a reduction in over-fitting, compared to an ANN without pre-training. Fine-tuning phase of the whole deep model is needed to ensure good results, as was shown by the comparison of TM-1 and TM-2. However, TM-1 obtained result is not overwhelming compared to deep learning improvements in other tasks [11], but it is a promising preliminary result. The low dimensionality of the task (univariate time-series forecasting using at most 96 inputs), and the smoothness of the indoor temperature time-series, reduce the benefit of using SDAE models, as was stated in [11]. Although RBMs and other deep techniques are being applied for time series analysis [12,13], as far as we know, the approach proposed in this paper has not been employed yet. Hence, a future work might be to use a larger forecasting input window, combined with multivariate forecasting (adding additional inputs as time, sun irradiance, and/or humidity), where higher dimensional dependencies are expected to be learned by deep learning techniques.

References

1. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting 14(1), 35–62 (1998)
2. Ben Taieb, S., Bontempi, G., Atiya, A., Sorjamaa, A.: A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Systems with Applications (2012) (preprint)
3. Zamora-Martínez, F., Romeu, P., Pardo, J., Tormo, D.: Some empirical evaluations of a temperature forecasting module based on Artificial Neural Networks for a domotic home environment. In: IC3K – KDIR (2012)
4. Ferreira, P., Ruano, A., Silva, S., Conceição, E.: Neural networks based predictive control for thermal comfort and energy savings in public buildings. Energy and Buildings 55, 238–251 (2012)
5. Utgoff, P.E., Stracuzzi, D.J.: Many-layered learning. Neural Comput. 14(10), 2497–2529 (2002)
6. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. 1(4), 541–551 (1989)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research 9, 249–256 (2010)
8. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11, 625–660 (2010)
9. Erhan, D., Manzagol, P.A., Bengio, Y., Bengio, S., Vincent, P.: The difficulty of training deep architectures and the effect of unsupervised pre-training. Journal of Machine Learning Research 5, 153–160 (2009)
10. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
11. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. 11, 3371–3408 (2010)
12. Chao, J., Shen, F., Zhao, J.: Forecasting exchange rate with deep belief networks. In: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 1259–1266 (2011)
13. Kuremoto, T., Kimura, S., Kobayashi, K., Obayashi, M.: Time Series Forecasting Using Restricted Boltzmann Machine. In: Huang, D.-S., Gupta, P., Zhang, X., Premaratne, P. (eds.) ICIC 2012. CCIS, vol. 304, pp. 17–22. Springer, Heidelberg (2012)
14. Cheng, H., Tan, P., Gao, J., Scripps, J.: Multistep-ahead time series prediction. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 765–774. Springer, Heidelberg (2006)
15. Bergstra, J., Desjardins, G., Lamblin, P., Bengio, Y.: Quadratic polynomials learn better image features. Technical Report 1337, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal (April 2009)
16. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305 (2012)
17. Zamora-Martínez, F., et al.: April-ANN toolkit, A Pattern Recognizer In Lua, Artificial Neural Networks module (2013), `https://github.com/pakozm/april-ann`
18. Taylor, J.: Exponential smoothing with a damped multiplicative trend. International Journal of Forecasting 19, 715–725 (2003)

Recurrent Fuzzy-Neural Network with Fast Learning Algorithm for Predictive Control

Yancho Todorov[1], Margarita Terzyiska[2], and Michail Petrov[2]

[1] Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences,
Acad. G. Bontchev st., bl. 2, 1113, Sofia, Bulgaria
yancho.todorov@iit.bas.bg
[2] Technical University-Sofia, Branch Plovdiv,
25, Tsanko Dustabanov St., 4000, Plovdiv, Bulgaria,
terzyiska@gmail.com, mpetrov@tu-plovdiv.bg

Abstract. This paper presents a Takagi-Sugeno type recurrent fuzzy-neural network with a global feedback. To improve the predictions and to minimize the possible model oscillations, a hybrid learning procedure based on Gradient descent and the fast converging Gauss-Newton algorithms, is designed. The model performance is evaluated in prediction of two chaotic time series – Mackey-Glass and Rossler. The proposed recurrent fuzzy-neural network is coupled with analytical optimization approach in a Model Predictive Control scheme. The potentials of the obtained predictive controller are demonstrated by simulation experiments to control a nonlinear Continuous Stirred Tank Reactor.

Keywords: recurrent fuzzy-neural networks, Takagi-Sugeno, predictive control, optimization, Gradient descent, Gauss-Newton method, momentum learning.

1 Introduction

An important step in nonlinear control is the development of a nonlinear model. Since, the most dynamical systems are complex and nonlinear, the task of selecting a good model structure and performing identification and control imposes the continuous research in this area.

In recent years, computational-intelligence techniques, such as neural networks, fuzzy logic and combined fuzzy-neural networks have become very popular and effective tools for identification and control of nonlinear plants. The problem of identification consists of choosing an appropriate model and adjusting the parameters such that the response of the model approximates the response of the real system to the same input. For this purpose, a general fuzzy-neural network (FNN) approach based on multiple *Linear Time Invariant* models (as ARMAX and e.t.c.) around various function points has been proposed. Fuzzy-neural networks are well known as a universal aproximators and different structures are developed for solving identification and control problems, but the most widely used FNN is the

© Springer-Verlag Berlin Heidelberg 2013

Takagi-Sugeno (TS) inference. The TS approach is a convex polytopic representation, which can be obtained either through mathematical transformation or through achieved linearization around various operating points. The main advantage of the TS model is the soft transition through any operating regions [1-2].

Many recent developments show that recurrent fuzzy-neural networks (RFNN) are more suitable in describing complicate dynamical systems than the FNN, because they can handle the time-varying inputs or outputs through its own natural temporal operation. RFNNs have an internal feedback loop that allows them to capture the dynamic response of a system with external feedback through delays. Due to its dynamic characteristic and relatively simple architecture, the RFNNs are useful tools for most real-time applications, such as predictive controllers. Stable predictive controller based on RFNN model is described in [3]. In [4-5] are presented predictive controllers based on RFNN where the feedback is after the fuzzyfication layer. A novel Type-2 RFNNs with asymmetric membership functions are proposed in [6-7]. TSK-type RFNNs for modeling and control are described in [8-9]. These RFNNs have additional inputs which increase the number of the fuzzy rules and the computational burden, respectively.

This motivates us to propose a simple RFNN using a TS inference mechanism with global feedback and a hybrid learning algorithm, combining the Newton and the Gradient descent methods. The Newton approach is applied in notion to the fuzzy rules consequents and provides a treat quickly convergence of the learning, while the Gradient descent is used to schedule the rule premises in order to avoid potential model oscillations. The proposed model structure is evaluated in modeling of two common chaotic time series, as well as its potentials into nonlinear model predictive control (MPC) scheme of a Continuous Stirred Tank Reactor (CSTR), are also studied by simulation experiments.

2 Recurrent Takagi-Sugeno Fuzzy-Neural Network

Since the middle of the 1980s, TS fuzzy-neural models have attracted a great deal of attention from industrial practitioners and academic researchers, especially because they can effectively approximate a wide class of nonlinear systems. Thus, in discrete time by using the NARX representation model (*Nonlinear AutoregRessive model with eXogenous inputs*) can be derived:

$$y(k) = f_y(x(k)) \tag{1}$$

where the unknown nonlinear function f_y can be approximated by Takagi-Sugeno type fuzzy rules:

$$\begin{aligned} R^{(i)}: if\ x_1\ is\ \tilde{A}_1^{(i)}\ and\ x_p\ is\ \tilde{A}_p^{(i)}\ then\quad & f_y^{(i)}(k) = a_1^{(i)} y_m(k-1) + a_2^{(i)} y_m(k-2) + ... \\ & + a_{ny}^{(i)} y_m(k-n_y) + b_1^{(i)} u(k) + b_2^{(i)} u(k-1) + ... + b_{nu}^{(i)} u(k-n_u) + b_0^{(i)} \end{aligned} \tag{2}$$

where *(i)=1,2,...N* denotes the number of the fuzzy rules $R^{(i)}$. A_i is an activated fuzzy set defined in the universe of discourse of the input x_i and the crisp coefficients a_1,

$a_2,\dots a_{ny}$, b_1, $b_2,\dots b_{nu}$ are the coefficients into the Sugeno function f_y. The input vector $\boldsymbol{x}$ contains regressors in notion the input/output history dependence. On Fig. 1 is shown the schematic diagram of the proposed recurrent TS fuzzy-neural network.

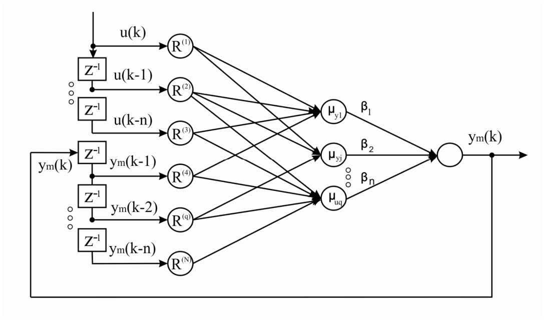


Fig. 1. Schematic diagram of the proposed recurrent fuzzy-neural network

The identification of the proposed recurrent network requires the two main groups of unknown parameters to be determined: the number of membership functions, their shape and the parameters of the function f_y in the consequent part of the rules. For this purpose, in this work a simplified fuzzy-neural approach is applied [10-11].

2.1 Learning Algorithm for the Designed Recurrent TS Fuzzy Neural Network

A two step learning procedure based on minimization of an instant error measurement function $E=\varepsilon^2/2$ and $\varepsilon(k)=y(k)-y_m(k)$ between the process output and the model output, is implemented. During the learning process, two groups of parameters in the fuzzy-neural architecture – premise and consequent parameters are under adaptation. The consequent parameters are the coefficients a_1, $a_2,\dots a_{ny}$, b_1, $b_2,\dots b_{nu}$ in the Sugeno function f_y and they are calculated by the following equations:

$$\beta_{ij}(k+1)=\beta_{ij}(k)+\eta(y-y_M)\bar{\mu}_y^{(j)}(k)x_i(k),\quad \beta_{0j}(k+1)=\beta_{0j}(k)+\eta(y-y_M)\bar{\mu}_y^{(j)}(k) \tag{3}$$

Where η is the learning rate and β_{ij} is an adjustable i^{-th} coefficient (a_i or b_i) in the Sugeno function f_y of the j^{-th} activated rule. The premise parameters are the centre c_{ij} and the deviation σ_{ij} of a Gaussian fuzzy set defined as:

$$\mu_{ij}^{(i)}(x_i)=\exp\left(-(x_i-c_i)/2\sigma_i\right)^2 \tag{4}$$

The parameters of a fuzzy set are calculated by using the following equations:

$$\begin{gathered}c_{ij}(k+1)=c_{ij}(k)+\eta(y-y_M)\bar{\mu}_y^{(j)}(k)[f_y^{(i)}-\hat{y}(k)]\frac{[x_i(k)-c_{ij}(k)]}{c_{ij}^2(k)}\\ \sigma_{ij}(k+1)=\sigma_{ij}(k)+\eta(y-y_M)\bar{\mu}_y^{(j)}(k)[f_y^{(i)}-\hat{y}(k)]\frac{[x_i(k)-\sigma_{ij}(k)]^3}{\sigma_{ij}^2(k)}\end{gathered} \tag{5}$$

2.2 Gauss-Newton Method for Hybrid Learning of the Network Parameters

To improve the efficiency of the proposed recurrent fuzzy-neural network, a Gauss-Newton method for adjusting the rules consequent parameters, is applied. Since the Newton method requires the computation of the second order derivative of the defined error cost term, taking into account (3) it can be rewritten:

$$\Delta\beta = -\left[\nabla^2 E(\beta)\right]^{-1} \nabla E(\beta) \tag{6}$$

The Hessian and the Gradient of *E(β)* are expressed as:

$$\nabla E(\beta) = J^T(\beta)e(k), \quad \nabla^2 E(\beta) = J^T(\beta)J(\beta) + \sum\nolimits_{j=1}^{N} e_j(k)\nabla^2 e_j(k) \tag{7}$$

where the dimension of the *Jacobian* matrix is *(NxN$_p$)*; *N*- the number of the training samples and N_p- is the number of adjustable parameters in the network. Using the Newton approach the second term in (7) is assumed equal to zero. Therefore, the update rule, according to (6) became*:*

$$\Delta\beta = -\left[J^T(\beta)J(\beta)\right]^{-1} J^T(\beta)e(k) \tag{8}$$

where the *Jacobian* according to adjustable parameters is calculated as:

$$J^T(\beta_{0j}) = \bar{\mu}_y{}^{(j)},\ J(\beta_{0j}) = \left[\bar{\mu}_y{}^{(j)}\right]^T,\ J^T(\beta_{ij}) = \bar{\mu}_y{}^{(j)}x_i,\ J(\beta_{ij}) = \left[\bar{\mu}_y{}^{(j)}x_i\right]^T \tag{9}$$

Finally, the recurrent equations for the rule consequent parameters are derived as follows:

$$\begin{aligned} \beta_{ij}(k+1) &= \beta_{ij}(k) + \eta\left[\bar{\mu}_y{}^{(j)}x_i(k)\left(\bar{\mu}_y{}^{(j)}x_i(k)\right)^T\right]^{-1}(y-y_M)\bar{\mu}_y{}^{(j)} + \zeta(\beta_{ij}(k)-\beta_{ij}(k-1)) \\ \beta_{0j}(k+1) &= \beta_{0j}(k) + \eta\left[\bar{\mu}_y{}^{(j)}\left(\bar{\mu}_y{}^{(j)}\right)^T\right]^{-1}(y-y_M)\bar{\mu}_y{}^{(j)}(k) + \zeta(\beta_{0j}(k)-\beta_{0j}(k-1)) \end{aligned} \tag{10}$$

where the second term represents an introduced momentum ζ in notion to previous increment of the adjusted parameter.

3 Model Predictive Control Policy

Using the designed recurrent TS fuzzy-neural model, the *Optimization Algorithm* computes the future control actions at each sampling period, by minimizing the following cost function:

$$J(k,u(k)) = \sum\nolimits_{i=1}^{N_2}(r(k+i)-\hat{y}(k+i))^2 + \rho\sum\nolimits_{i=1}^{N_u}\Delta u(k+i-1)^2 \tag{11}$$

where $\hat{y}$ is the predicted model output, *r* is the reference signal and *u* is the control action. The tuning parameters of the stated predictive controller are: N_1, N_2, N_u and ρ.

N_1 and N_2 are the minimum/maximum prediction horizons, N_u is the control horizon and ρ is the weighting factor penalizing changes in the control actions. Since, the criterion function is a quadratic one and there are no imposed constraints on the control action, the minimization procedure is performed iteratively. If the criterion J is minimized with respect to the future control moves u, then their optimal values can be calculated by applying the condition $\nabla J[k, U(k)]=0$. Thus, each element from the gradient vector is calculated using the following equation:

$$\frac{\partial J[k,U(k)]}{\partial U(k)} = \left[-2[R(k)-\hat{Y}(k)]^T \frac{\partial \hat{Y}(k)}{\partial U(k)} + 2\rho \hat{U}(k)^T \frac{\partial \hat{U}(k)}{\partial U(k)} \right] \tag{12}$$

It can be seen, that two partial derivatives have to be determined. The first one is $\partial\hat{Y}(k)/\partial U(k)$, and second one is $\partial\hat{U}(k)/\partial U(k)$. Each element from the first group of partial derivatives is calculated by the following equations [10-12]:

$$\frac{\partial \hat{y}(k)}{\partial u(k)} = \sum_{i=1}^{N} b_1^{(i)} \bar{\mu}_y^{(i)}(k) \;.....\; \frac{\partial \hat{y}(k+N_2)}{\partial u(k)} = \sum_{i=1}^{N} \begin{bmatrix} a_1^{(i)} \dfrac{\partial \hat{y}(k+N_2-1)}{\partial u(k)} + \dots \\ +a_2^{(i)} \dfrac{\partial \hat{y}(k+N_2-2)}{\partial u(k)} \end{bmatrix} \bar{\mu}_y^{(i)}(k+N_2) \tag{13}$$

Since, $\Delta u(k)=u(k)-u(k-1)$ the $\partial\hat{U}(k)/\partial U(k)$ represents matrix of zeros and ones. Thus, the recurrent equations for calculation of the control actions along the horizon are derived as:

$$\Delta u(k+N_u-1) = \rho^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-1)} + ... + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-1)} \right] \tag{14}$$

$$\Delta u(k) = \Delta u(k+1) + \rho^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k)} + ... + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k)} \right] \tag{15}$$

4 Results and Discussion

4.1 Recurrent TS Model Evaluation by Prediction of Chaotic Time Series

Chaos is a common dynamical phenomenon in various fields [13] and different definitions as series representations exist. Chaotic time series are inherently nonlinear, sensitive to initial conditions and difficult to be predicted. For that purpose, the chaotic time series prediction based on measurement is a practical technique for studying characteristics of complicated dynamics [14] and evaluation of the accuracy of different types of nonlinear models as RNN's. In this study, a two chaotic time series, Mackey-Glass [15] and Rossler [16] are used to assess the performance prediction of the proposed recurrent TS network, with chosen fixed momentum of ζ=0.098. On Fig. 2 is demonstrated the model performance in prediction of the Mickey-Glass chaotic times series, with the following parameters: a=0.2; b=0.1; C=10; initial conditions x_0=0.1 and τ= 17s. As it can be seen, the proposed model structure predicts accurately the generated time series, with minimum prediction error and fast transient response of the RMSE, reaching values closer to zero.

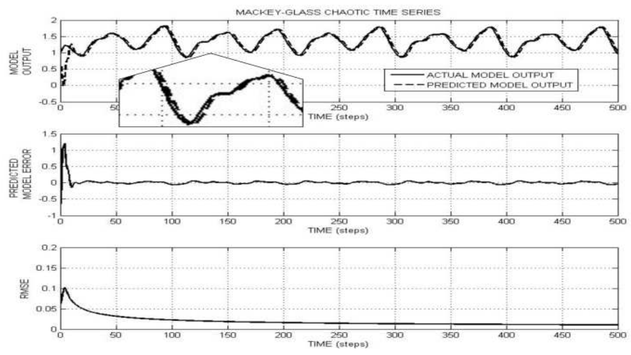


Fig. 2. Model validation by using Mackey-Glass time series

On Fig. 3 are shown the obtained results in case of Rossler chaotic series prediction with the following parameters: a=0.2; b=0.4; c=5.7; initial conditions x_0=0.1; y_0=0.1; z_0=0.1. The obtained results show again a good model performance with minimum error prediction and fast transient response of the RMSE, approaching to zero.

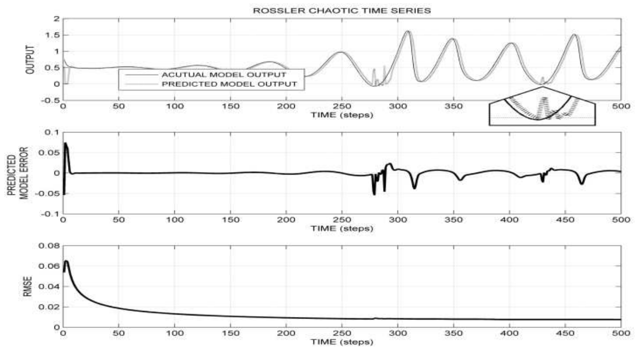


Fig. 3. Model validation by using Rossler time series

4.2 Evaluation of the Proposed MPC with Recurrent TS Network

To evaluate the performance of the proposed MPC control policy a simplified model of nonlinear CSTR plant is used for simulation experiments. The dynamic equations of the nonlinear plant are given by:

$$\dot{x}_1 = -x_1 + D_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\phi}\right),\ \dot{x}_2 = -(1+\delta)x_2 + BD_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\phi}\right) + \delta u \qquad (16)$$

where x_1 and x_2 represent the dimensionless reactant concentration and the reactor temperature, respectively. The control action *u* is dimensionless cooling jacket temperature. The physical parameters in the CSTR model equations are D_a, ϕ, B and δ which correspond to the Damkhler number, the activated energy, the heat of reaction and the heat transfer coefficient, respectively. Based on the nominal values of the system parameters, D_a=0.072, ϕ =20, B=8 and δ=0.69, the open-loop CSTR exhibits three steady states $(x_1,x_2)_A$=(0.144, 0.886), $(x_1,x_2)_B$=(0.445, 2.75) and $(x_1,x_2)_C$=(0.765, 4.705), where the upper and the lower steady states are stable, whereas the middle one is unstable. The control objective here is to bring the nonlinear CSTR from the stable equilibrium point $(x_1,x_2)_A$ to the unstable one $(x_1,x_2)_B$ [17]. On the Fig. 4 are demonstrated the obtained results of CSTR control using the recurrent TS fuzzy-neural network and different values of the penalty term ρ, as well as a comparison is made with a predictive controller using the classical TS representation (non-recurrent NARX representation). Both models have been coupled with a Gradient descent algorithm as optimization approach into a MPC with the following set of horizons, N_1=1, N_2=5 and N_u=3. As it can be seen, varying the penalty term when using the recurrent TS implementation, leads to smooth controller operation with small system overshoot. On the other hand, the performed comparative study on equal initial conditions for the models show that the assumed recurrent TS representation leads to more faster transient response of the system. Taking into account that, the most of the real time processes under control, have a smooth nature and the instant samples for model operation are relatively high, it can be concluded that the proposed recurrent fuzzy-neural model may be a promising solution in a MPC scheme.

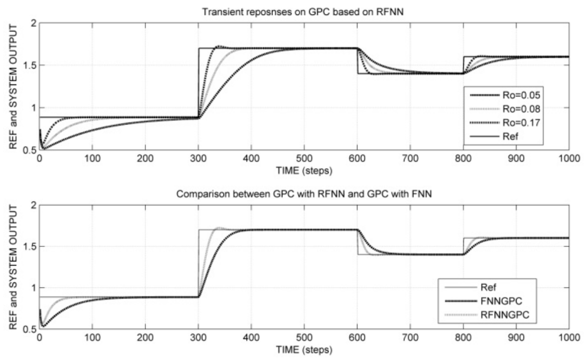


Fig. 4. Process responses in a MPC scheme compared to classical TS model

Conclusions: It was presented in this paper a recurrent implementation of a Takagi-Sugeno fuzzy-neural network with a global feedback. The model performance has been evaluated in prediction of two commonly used chaotic times series – Mackey-Glass and Rossler. The obtained results show a minimal error prediction of the chaotic series of different frequency and amplitude. The operation of the proposed model is also studied into a MPC control scheme for a CSTR and compared to classical TS non-recurrent implementation, on equal initial conditions. The results show a faster controller response when using recurrent TS model with minimal overshoot in case of changing reference. This makes the adopted modeling approach a promising solution for designing different predictive controllers.

Aknowlegment. The research work reported in the paper is partly supported by the project ***AComIn*** "Advanced Computing for Innovation", grant 316087, funded by the FP7 Capacity Programme (Research Potential of Convergence Regions).

References

1. Mendes, J.: Adaptive fuzzy generalized predictive control based on Discrete-Time T-S fuzzy model. In: Proc. IEEE Conf. of Em. Tech. and Factory Automation, pp. 1–8 (2010)
2. Chaladi, M., Borne, P.: Multiple Models Approach in Automation: TS Fuzzy Systems. Wiley (2012)
3. Chi-Huang, L., Chi-Ming, L.: Stable predictive control based on recurrent fuzzy neural networks. In: Proceeding on 8th Asian Control Conf (ASCC), pp. 897–901 (2011)
4. Chi-Huang, L., Ching-Chih, T.: Generalized predictive control using recurrent fuzzy neural networks for industrial processes. Journal of Process Control 17, 83–92 (2007)
5. Mendes, J., Sousa, N., Araujo, R.: Adaptive predictive control with recurrent fuzzy neural network for industrial processes. In: 16th IEEE ETFA Conference, pp. 1–8 (2011)
6. Ching-Hung, L., Chang, H., Ting Kuo, C., Chieh Chien, J., Wei Hu, T.: A Novel Recurrent Interval Type-2 Fuzzy neural Network for Nonlinear Channel Equilization. In: Proceeding of the Int. MultiConf. of Eng. and Computer Sci., pp. 7–12 (2009)
7. Chia-Feng, J., Yang-Yin, L., I-Fang, C.: Dynamic System Identification Using A Type-2 Recurrent Fuzzy Neural Network. In: Proc. of the 7th Asian Control Conference (2009)
8. Chia-Feng, J., Chun-Feng, L., Po-Han, C.: Dynamic Plant Control Using Recurrent Fuzzy Controller with Ant Colony Optimization in Real Space. In: Proc. of the International IEEE Conference on Systems Man and Cybernetics, pp. 1134–1138 (2010)
9. Ya-Ling, C., Ching-Chih, T.: A TSK-Type Recurrent Fuzzy Neural Network Adaptive Inverse Modeling Control for a Class of Nonlinear Discrete-Time Time-Delay Systems. In: Proc. of SICE Annual Conference, pp. 2390–2393 (2010)
10. Terzyiska, M., Todorov, Y., Mitev, A., Petrov, M.: Nonlinear Model Based Predictive Controller using a Fuzzy-Neural Hammerstein model. In: Proc. of the Int. Conf. 'Modern Trends in Control', pp. 299–308 (2006)
11. Terzyiska, M., Todorov, Y., Petrov, M.: Nonlinear Model Predictive Controller with Adaptive Learning rate Scheduling of an internal model. In: Proc. of the Int. Conf. 'Modern Trends in Control', pp. 289–298 (2006)
12. Todorov, Y., Tsvetkov, T.: Volterra Model Predictive Control of a Lyophilization plant. In: Proc. of the 4th IEEE Conf. "Intelligent Systems", vol. 3, pp. 13–18 (2008)
13. Yao, J., Mao, J., Zhang, W.: Application of Fuzzy Tree on Chaotic Time Series Prediction. In: IEEE Proc. of Int. Conf. on Aut. and Logistics, pp. 326–330 (2008)
14. Lai, Y.: Recent developments in chaotic time series analysis. International Journal of Bifurcation and Chaos 13(6), 1383–1422 (2003)
15. Diaconescu, E.: The use of NARX Neural Networks to predict Chaotic Time Series. WSEAS Trans. on Computer Research 3(3), 182–191 (2008)
16. Archana, R., Unnikrishnan, A., Gopikakumari, R.: Bifurcation Analysis of Chaotic Systems using a Model Built on Artificial Neural Networks. In: Proc. of Int. Conf. on Comp. Tech. and Artificial Intelligence, pp. 198–202 (2013)
17. Ray, W.H.: Advanced Proceess Control. McGraw-Hill, New York (1981)

Real-Time Interface Board for Closed-Loop Robotic Tasks on the SpiNNaker Neural Computing System

Christian Denk[1], Francisco Llobet-Blandino[1], Francesco Galluppi[2], Luis A. Plana[2], Steve Furber[2], and Jörg Conradt[1]

[1] Fachgebiet Neurowissenschaftliche Systemtheorie, Fakultät für Elektro- und Informationstechnik, Technische Universität München, 80290 München, Germany
{christian.denk,llobetblandino,conradt}@tum.de
[2] Advanced Processor Technologies Group, School of Computer Science, University of Manchester, Manchester M13 9PL, UK
{francesco.galluppi,plana,sfurber}@cs.man.ac.uk

Abstract. Various custom hardware solutions for simulation of neural circuitry have recently been developed, each focusing on particular aspects such as low power operation, high computation speed, or biologically detailed simulations. The SpiNNaker computing system has been developed to simulate large spiking neural circuits in real-time in a network of parallel operating microcontrollers, interconnected by a high-speed asynchronous interface. A potential application area is autonomous mobile robotics, which would tremendously benefit from on-board simulations of networks of tens of thousands of spiking neurons in real-time. Currently, the SpiNNaker hardware circuit boards provide a single Ethernet interface for booting, debug, and input and output of data, which results in a severe bottleneck for sensory perception and motor control signals. This paper describes a small and flexible real-time I/O-hardware interface to connect external devices such as robotic sensors and actuators directly to the fast asynchronous internal communication infrastructure of the SpiNNaker neural computing system. We evaluate performance in terms of package throughput and present a simple application demonstration of a closed loop mobile robot interpreting visual data to approach the most salient stimulus.

Keywords: massively-parallel simulation of spiking neurons, SpiNNaker, hardware interface board, mobile robotics.

1 Introduction and Related Work

In computational neuroscience research various customized computing systems for the simulation of neuronal networks are under development, such as Facets/ BrainScales/ HBP [1], Neuro-grid [2], dedicated aVLSI computing chips [3], or the SpiNNaker spiking network computing system [4]. All such hardware resembles brain style information processing, which in contrast to traditional general purpose computers offers various advantages, e.g. reduced power consumption or increased

© Springer-Verlag Berlin Heidelberg 2013

processing speed for elaborate or complex neural models. Most larger systems are designed for “neuronal number crunching” (i.e. detailed neuronal modeling) and exist in a closed computer rack with well controlled digital input and output channels. At the other side of the spectrum, there are small scale test-case implementations of neuromorphic functionality. Only very few such neuromorphic computing systems have yet operated in real time on noisily perceived sensory data and produced reasonable motor outputs to interact with the surrounding environment.

Engineers and robotics systems designers however can strongly benefit from flexible instantiations of real-time neural information processing algorithms. Many robotic research groups agree that the neuronal style of information processing is advantageous for real time sensory processing and motor control, as this is what the brain and especially cortex is largely devoted to do. Current applications of neuromorphic hardware in closed loop systems typically keep a desktop computer in the loop to acquire sensory data from a robot, translate it into “neural” form, and provide it to the neuromorphic hardware and vice versa for motor output. Such a setup faces various inherent drawbacks: (a) large size and power consumption, which defies application on most mobile robots; (b) processing delays that might break real-time control loops; (c) no autonomy, which limits the operation range as a connection to a stationary computer is needed and (d) waste of resources, as the computer often only “translates” data.

In this paper we present a standalone interface solution for a direct connection of various types of external hardware to the SpiNNaker [4] neural computing system, and present an application example that is autonomously executed in real-time on-board a mobile robot. The developed board is compact in size, supports high data transfer rates, requires little operating power (thereby allowing extended runtime on batteries), operates autonomously, offers various connection options for existing robots and sensors, and provides simple customizable extensions to additional sensors and actuators (or other robots).

2 An Autonomous Mobile Robot with a Neuronal Computing System

2.1 The SpiNNaker Neural Network Computing System

The SpiNNaker computing system [4] is designed for massively-parallel computations of spiking neural networks. Each SpiNNaker chip consists of 16+2 generic ARM968 cores, a shared 128MByte SDRAM module, and an asynchronous high-speed communication interface with six bi-directional links. The chips execute arbitrary code in each of the 16 user accessible cores, but the overall system design is optimized for simulations of large spiking neural networks (e.g. LIF or Izhikevich neurons). Current SpiNNaker systems offer 4 chips (64 cores) or 48 chips (768 cores) with each core simulating up to 1000 neurons in real time [5], thereby allowing networks of 64.000 (768.000) spiking neurons. The fast asynchronous communication interface is designed to route neural action potentials from arbitrary neurons to a large number of other neurons. Various options for neural network implementations exist, from API library calls to interpreters of neural description languages such as PyNN [6] or NeNGO[7].

2.2 The Holonomic Mobile Robot Platform

The mobile robot used in this project (Figure 1, left) is an omni-directional platform of 26cm diameter, with embedded low-level motor control and elementary sensors. An embedded microcontroller obtains motion commands in x and y direction and rotation through a UART interface, and continuously adapts three motor signals to maintain requested velocities. The robot's on-board sensors include wheel encoders, a 9 Degrees of Freedom (DOF) inertial measurement unit and a simple bump-sensor ring to trigger binary contact switches upon contact with objects.

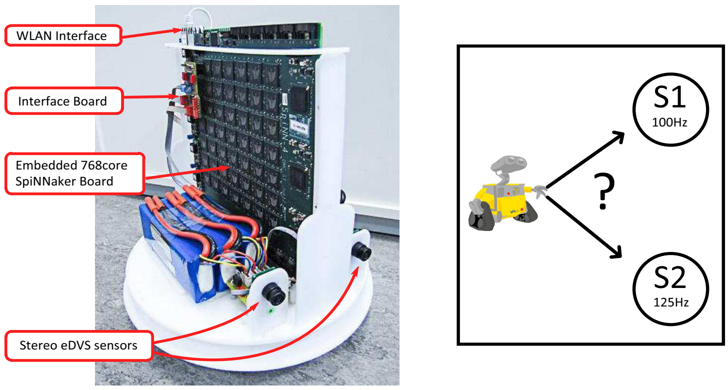


Fig. 1. Left: Autonomous mobile robot with on-board vision sensors, SpiNNaker hardware and interface board. Right: exemplary robot task: select the strongest out of multiple visual stimuli.

2.3 The Embedded Dynamic Vision Sensor

The dynamic vision sensor (DVS) [9] used as spiking sensory input in this project is an address-event silicon retina that responds to temporal contrast. Each output spike represents a quantized change of log intensity at a particular pixel since the last event from that pixel. All 128x128 pixels operate asynchronously and signal illumination changes within a few microseconds after occurrence. We developed an embedded DVS system (eDVS) [10] composed of a DVS chip connected to an ARM7 microcontroller that initializes the DVS and captures events. In this project the microcontroller streams all obtained events over a UART port into the SpiNNaker interface board (Section 3).

3 Design and Specifications of the Real-Time Interface Board

Distributed computing cores in SpiNNaker exchange data on an energy and speed efficient asynchronous interface, which unfortunately is tedious to connect to external hardware. This section presents our developed interface board that attaches to this interface to send and receive native SpiNNaker packets, and to act as customizable interpreter between external hardware and the SpiNNaker system.

3.1 The SpiNNaker Inter-Chip Communication Protocol and Interface

Messages in the SpiNNaker system (typically neuronal "spikes") are transferred as 40-bit packets, composed of an 8-bit header and a 32-bit routing key. Packets can also carry an optional 32-bit payload. [5]. Communication between chips happens on two unidirectional asynchronous interfaces composed of 7 data lines (plus acknowledge), which are optimized for low energy consumption and fast data transfer rates: static levels on the data lines are meaningless, only transitions of bits encode a value. Any double bit flip on those 7 lines encodes the next nibble of the 40/72bits data word ("2-of-7 code"), which needs to be acknowledged by toggling the signal on the respective acknowledge line. This protocol is fast (allows up to 6M packets per second) and energy efficient, but is difficult to implement for existing sensors and/or mobile robots. Hence we developed a generic interface board that on one side follows the communication protocol enforced in SpiNNaker, and on the other side offers various generic options to connect external hardware.

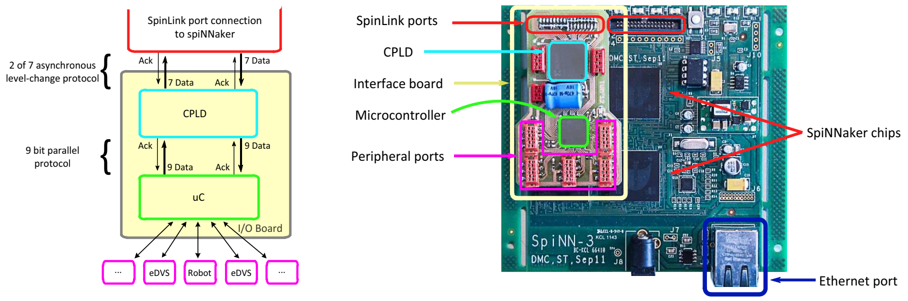


Fig. 2. Left: Sketch of information flow SpiNNaker ↔ external hardware (e.g. eDVS, robot). Right: SpiNN-3 system (4 chip board) with attached interface board.

3.2 The Developed SpiNNaker Interface Board

The Interface Board receives and transmits SpiNNaker data packets in the 2-of-7 bit toggling format on one side (Figure 2, red connectors) and offers a variety of flexible interfaces for sensors and/or actuators on the other side (Figure 2, purple connectors). A fast on-board microcontroller (STM 32F407, 32bit ARM Cortex M4, 168MHz; Figure 2, green) allows flexible customization of translation protocols between SpiNNaker packets and sensor or actuators signals as described below. For efficiency we added a CPLD (XILINX Coolrunner-II, XC2C64A; Figure 2, blue) in the communication path, which translates between 2-of-7 bit-toggling codes and 9 bit data bus level signals for the microcontroller (8 data bits and 1 handshaking signal). All communication (SpiNNaker ↔ CPLD ↔ microcontroller) in both directions generates appropriate handshake signals to guarantee lossless transmission of data.

The microcontroller consecutively retrieves all available data from SpiNNaker and connected peripherals and translates the data into the respective other format. After the translation, the data are forwarded to the respective devices as soon as possible (e.g. the SpiNNaker and/or UART transmit ports free).

The presented interface board is easy to extend for upcoming system demands, even for users inexperienced with electronic hardware and/or microcontroller programming: the main-loop that continuously processes data is essentially a large lookup-table, which makes it easy to include different sensors and actuators without the need to be aware of SpiNNaker low-level programming (such as "2-of-7 bit toggling") or routines to communicate over UART, SPI, or TWI.

The developed interface board allows neural models running on SpiNNaker to receive sensory input signals and to control actuators. Performance evaluation in terms of number of sent / received packages is shown in Figure 3, left.

4 Application: Winner-Takes-All Network on Mobile Robot

We demonstrate the SpiNNaker robot platform equipped with a forward pointing eDVS performing a simple closed loop robotic task: various visual stimuli (lights with different flashing frequencies) are positioned at some distance of the robot. The system should identify and approach the most active stimulus (Figure 1, right). In our demonstration we implement a nonlinear robust "Winner Takes All (WTA)" [8] (Figure 3 right) neuronal network on the visual input to identify the most salient stimulus. All elements of this network are spiking neurons, which is well suited for the SpiNNaker platform. We demonstrate and evaluate the performance of our implementation in a closed-loop experiment.

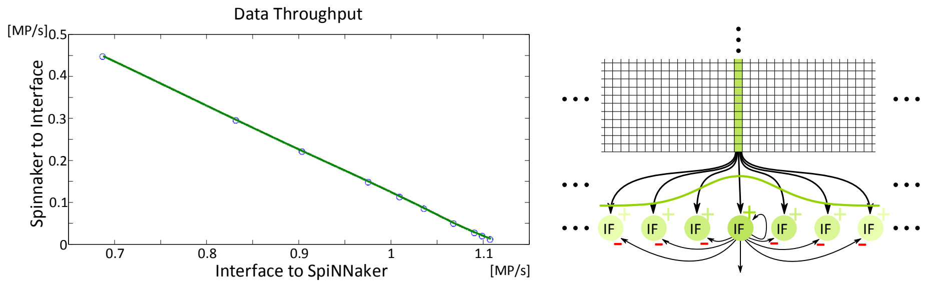


Fig. 3. Left: Transfer rates for simultaneous transmission and reception of SpiNNaker packets (32bit) in million packets per second. Right: Sketch of WTA network to process visual stimuli.

4.1 Winner Takes All Networks

The WTA network is a well-established computing principle [8] to sharpen spatially and/or temporally related diffuse input signals. A WTA network can be described as a robust max operation: instead of identifying the maximum of several inputs at each time step, a WTA filter is a dynamical system that selects the maximum over a sliding window in time and possibly space, implementing hysteresis and thereby generating robust output [8].

Our implementation of the WTA network uses a layer of Integrate and Fire (IF) neurons to which all eDVS events of one input column are propagated (Figure 3, right). These IF neurons compete for activation, but only the most active neuron will reach its firing threshold and thereby is identified network winner. Such firing of a winning neuron has two consequences: (a) inhibition of its competitors, and (b) resetting/initializing itself to a non-zero membrane potential (self-excitation). This recurrence generates the desired hysteresis as discussed in [8].

The sketch of a sub-region of the full 128-node network in Figure 3, right, depicts event propagation and WTA implementation: the top grid shows a part of the eDVS' pixel map. Our selection problem is essentially a one-dimensional task, as all pixels in a column (y) for any given row (x) support the same driving direction. This grouping results in a one-dimensional visual input vector of size 128, represented by the activity levels in designated neurons. We apply a spatial low-pass distribution on the input signal (green Gaussian in Figure 3, right) that produces strong excitation at the center neuron and symmetrically decayed excitation at its neighbors.

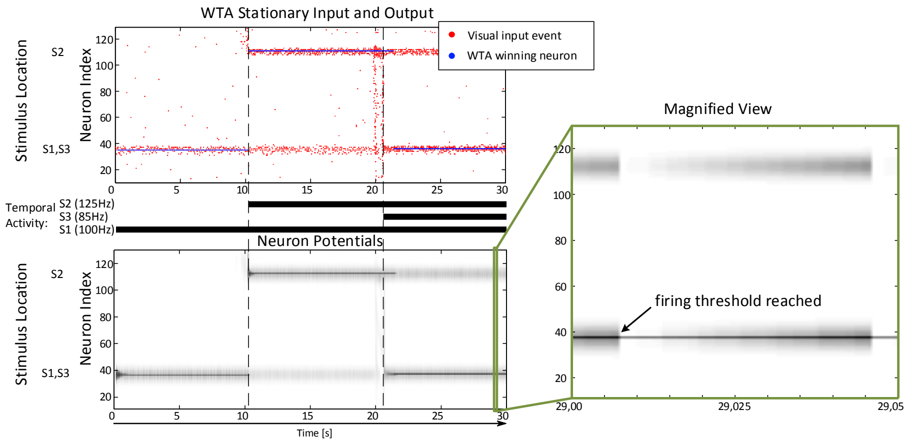


Fig. 4. Top Left: Stimulus activation (black bars) and corresponding evoked visual events over time (red dots). Bottom Left: activation of WTA neurons. Right: time magnified view.

4.2 WTA Implementation on SpiNNaker Hardware

The developed interface board (Section 3) translates and propagates all eDVS visual input events as native SpiNNaker packets, conveying the x-coordinate as source address. These packets are injected at the lower left chip on the SpiNNaker board, and distributed evenly among several other SpiNNaker chips and cores. For simplicity (as this is a proof-of-concept implementation) each core implements only a single IF neuron, who's potential is augmented by incoming events according to the Gaussian weighting function. Upon reaching firing threshold, an output spike causes all other distributed neurons to decrease their respective potentials (see Figure 4, right). Various dedicated motor neurons detect WTA output spikes and compute temporally low-pass filtered rate-encoded driving signals, which are sent through the SpiNNaker interface board to the robot.

4.3 Evaluation of the Demonstration System

We demonstrate our implementation of the WTA network in two different scenarios: (a) stationary with multiple different alternating stimuli and (b) on an autonomous robot driving towards partially occluded stimuli (Figure 1, right).

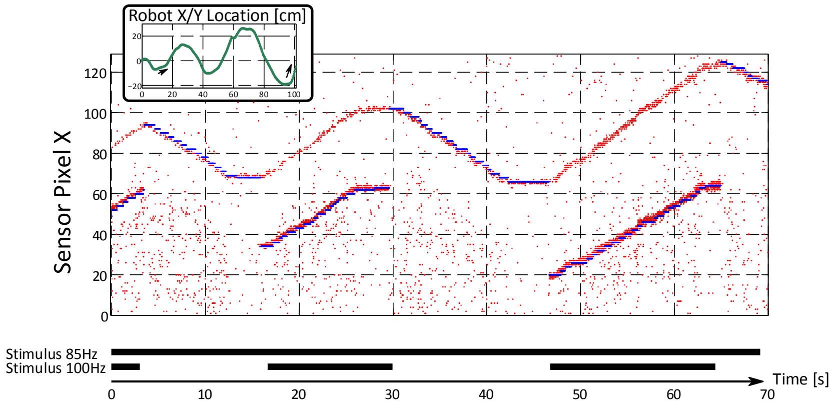


Fig. 5. Closed loop SpiNNaker-robot experiment: WTA alternatively identifying winner (blue) out of all stimuli (red dots); robot continuously approaches (and centers) the respective winner

For scenario (a), we provide three distinct LED stimuli (S1-3), each flashing with a particular frequency (see Figure 4, left). We position all stimuli so that the two low frequency stimuli are at roughly similar x coordinates (around x=38), whereas the most active stimulus is located elsewhere in the field-of-view (around x=110). The LED stimuli are turned on according to the timeline (black bar) in Figure 4. The upper graph shows input events (red dots) and WTA output spikes (blue dots). The lower graph displays WTA neuron integrator activation over time (darker parts indicate higher activation). Initially, with only S1 active, the WTA network identifies x=38 as center of activation, and provides a unique, spatially stable output firing despite background noise and a slightly broadened input signal distribution. After activation of S2 (with increased activity compared to S1) at t=10s, the WTA network transitions to this stimulus as winner; again identifying a spatially stable unique winning location. Additionally activating S3, which has the lowest frequency of all, but is spatially co-located with S1, yields a return to the first winning location, due to the Gaussian distribution of input signals to neurons, which allows the two less active stimuli to surpass the most active stimulus in sum. The magnified view (Figure 4, right) shows the neuronal activation over time; note the spatially distributed increase of activation around stimuli, and global inhibition and local self-excitation after a neuron fired.

In demonstration scenario (b), an autonomous mobile robot is controlled by the WTA output (Figures 1 and 5). The WTA network focuses on the most active stimulus, which causes the robot to continuously turn towards and approach that stimulus. In the closed loop system, the winning stimulus approaches the center of the vision sensor (pixel 64) as the robot turns (Figure 5; inset shows the robot's trajectory in top-down view). We repeatedly occlude the stronger stimulus (see activation bars in Figure 5), which produces alternating robot motion towards the stronger stimulus, thereby demonstrating switching behavior of the WTA network.

5 Results and Discussion

The SpiNNaker computing system provides a powerful and easy to learn neuronal computing infrastructure for computational modelers, which allows simulation of large scale spiking neural system in real-time. However, scenarios with real-time input/output currently require a PC in the loop or are custom developed FPGA chips for a particular piece of hardware, because of the SpiNNaker internal communication bus which is incompatible with existing sensors and/or robots.

In this project we presented a solution to flexibly interface various external hardware (such as sensors and/or robots) to the SpiNNaker computing system. The developed interface is small, allows high data transfer rates (sufficient even for visual data), and is easily customizable for future additional sensors and actuators without requiring in-depth knowledge about the SpiNNaker communication protocol. We demonstrated the performance of the developed system in two example settings: (a) stationary sensors with variable stimuli and (b) in an autonomous closed loop robotic experiment. The presented application shall be viewed as a proof-of-principle, not as an exhaustive evaluation of the board. In fact the demonstration only requires a small subset of the implemented features (e.g. significantly higher data rates are possible, refer to Figure 3, left). We are currently using the interface board in ongoing research such as a stereo optic flow processing and a neural model of grid cells for navigation.

References

1. Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M.A., Schmuker, M., Brüderle, D., Schemmel, J., Meier, K.: Six Networks on a Universal Neuromorphic Computing Substrate. Frontiers in Neuroscience 7 (2013)
2. Choudhary, S., Sloan, S., Fok, S., Neckar, A., Trautmann, E., Gao, P., Stewart, T., Eliasmith, C., Boahen, K.: Silicon Neurons That Compute. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part I. LNCS, vol. 7552, pp. 121–128. Springer, Heidelberg (2012)
3. Badoni, D., Giulioni, M., Dante, V., Del Giudice, P.: An Avlsi Recurrent Network of Spiking Neurons with Reconfigurable and Plastic Synapses. In: IEEE (ISCAS), pp. 1227–1230 (2006)
4. Khan, M., Lester, D., Plana, L.A., Rast, A., Jin, X., Painkras, E., Furber, S.B.: SpiNNaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor. In: IEEE International Joint Conference on Neural Networks (IJCNN), pp. 2849–2856. IEEE (2008)
5. Plana, L.A., Bainbridge, J., Furber, S., Salisbury, S., Shi, Y., Wu, J.: An on-Chip and Inter-Chip Communications Network for the SpiNNaker Massively-Parallel Neural Net Simulator. In: 2nd ACM/IEEE NoCS, pp. 215–216. IEEE (2008)
6. Galluppi, F., Davies, S., Rast, A., Sharp, T., Plana, L.A., Furber, S.: A Hierachical Configuration System for a Massively Parallel Neural Hardware Platform. In: Proceedings of the 9th conference on Computing Frontiers, pp. 183–192. ACM (2012)
7. Galluppi, F., Davies, S., Furber, S., Stewart, T., Eliasmith, C.: Real Time on-Chip Implementation of Dynamical Systems with Spiking Neurons. In: IJCNN, pp. 1–8. IEEE (2012)
8. Oster, M., Douglas, R., Liu, S.-C.: Computation with Spikes in a Winner-Take-All Network. Neural Computation 21, 2437–2465 (2009)
9. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128× 128 120dB 15ms Latency Asynchronous Temporal Contrast Vision Sensor. IEEE Solid-State Circuits 43, 566–576 (2008)
10. Conradt, J., Berner, R., Cook, M., Delbruck, T.: An Embedded AER Dynamic Vision Sensor for Low-Latency Pole Balancing. In: IEEE ECV, pp. 780–785. IEEE (2009)

A Software Framework for Cognition, Embodiment, Dynamics, and Autonomy in Robotics: *cedar*

Oliver Lomp,[1] Stephan Klaus Ulrich Zibner,[1] Mathis Richter,[1]
Iñaki Rañó,[2] and Gregor Schöner[1,⋆]

[1] Institut für Neuroinformatik, Ruhr-Universität Bochum
Universitätsstr. 150, 44780 Bochum, Germany
`{oliver.lomp,stephan.zibner,mathis.richter,`
`gregor.schoener}@ini.rub.de`
`http://www.ini.rub.de`

[2] Intelligent Systems Research Centre, University of Ulster
Northland Road, Derry, Northern Ireland BT48 J7L, UK
`i.rano@ulster.ac.uk`
`http://isrc.ulster.ac.uk`

Abstract. We present *cedar*, a software framework for the implementation and simulation of embodied cognitive models based on Dynamic Field Theory (DFT). DFT is a neurally inspired theoretical framework that integrates perception, action, and cognition. *cedar* captures the power of DFT in software by facilitating the process of software development for embodied cognitive systems, both artificial and as models of human cognition. In *cedar*, models can be designed through a graphical interface and interactively tuned. We demonstrate this by implementing an exemplary robotic architecture.

Keywords: software framework, embodied cognition, neural dynamics, Dynamic Field Theory, cognitive robotic models.

1 Introduction

As scientists from diverse fields recognize the critical importance of grounding cognitive function in sensory-motor processes, the embodiment stance is becoming a shared perspective in the study of both artificial and natural cognition [4]. Embracing embodiment has consequences for cognitive modeling. Models of human cognition that account for psychophysical or neural data must include motor control and the associated sensory processes. Artificial cognitive systems have to be implementable on robotic hardware so that intelligent behavior may be generated while the system is situated in the real world. This requires that cognitive

⋆ The authors acknowledge the financial support of the European Union Seventh Framework Programme FP7-ICT-2009-6 under Grant Agreement no. 270247—NeuralDynamics. This work reflects only the authors' views; the EC is not liable for any use that may be made of the information contained herein.

© Springer-Verlag Berlin Heidelberg 2013

models are linkable to real-time sensory inputs, capable of controlling effectors in real time, and must accommodate updating and control in a closed loop. Embodiment thus requires a theoretical framework suited to address these issues, and its demands are mirrored in its implementation. These demands include integration across sensory, motor, and cognitive processes, real-time linkage to sensors and effectors, fast prototyping, operation in closed loop, and tools for model evaluation.

This paper introduces *cedar*, a software framework for the design and evaluation of embodied cognitive systems that builds on *Dynamic Field Theory* (DFT). DFT is a neural dynamic theoretical framework for cognition [11] that is tailored to the embodiment paradigm. DFT has been used to model experimental data from psychology (see, e.g., [6,12]) as well as to design cognitive robotic architectures [3,9,14]. DFT uses dynamic neural fields (DNFs) as universal building blocks. A DNF represents a pattern of neural activation defined over continuous metrical dimensions (e.g., visual space, feature space). Neural activation evolves in time as described by a dynamical system (see Amari [1] for the mathematics). DNFs form stable *peaks* of activation as attractor states of the neural dynamics. These represent perceptual, motor, or cognitive decisions and emerge from bifurcations, in which non-peak solutions become unstable. The bifurcations demarcate different dynamic regimes of a DNF that reflect cognitive functions such as detection, selection, and working memory [11]. The stability properties of each DNF make it possible to build structurally complex architectures within DFT [14]. Their design requires that a relatively small (compared to conventional neural networks), but critical set of parameters be tuned to achieve the correct regimes in all DNFs under the desired environmental and task conditions.

We define the following requirements for an integrated software framework for DFT.

1. It must be possible to build architectures from common components, in particular, DNFs and their couplings, and to inspect these components visually.
2. Simulations must be real-time capable. Parameters must be changeable on-the-fly, which allows to inspect their impact on any component.
3. Components of an architecture must be connectable to physical hardware (sensors, effectors).

We take inspiration from software frameworks developed for related approaches to cognitive modeling that address different requirements. Parallel Distributed Processing [10], for instance, facilitates design through software such as the 'PDPTool' for Matlab, while other strands of connectionism are supported by tools implemented in C++ such as 'iqr' [2]. The large-scale spiking networks based on the Neural Engineering Framework [5] become practical through the efficient 'Nengo' simulator with its graphical user interface [13].

The integrated software framework *cedar* is our solution to fulfilling the requirements listed above. *cedar* is open-source, developed in C++, and available[1] for Linux, Mac OS, and Windows.

[1] *cedar* is available for download with extensive documentation at `http://cedar.ini.rub.de`

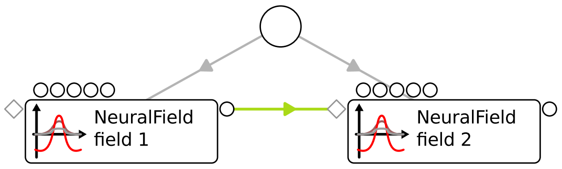


Fig. 1. *Graphical notation of the interfaces of cedar.* The figure contains two components ('field 1' and 'field 2') and a looped trigger (large circle).

2 A Software Framework for Dynamic Field Theory

DFT models consist of interconnected dynamical systems, most prominently dynamic neural fields (DNFs), that evolve continuously in time. The connections between DNFs vary in complexity, from simple identity mappings to chains of more complex operations that may include transformations such as mappings between DNFs of different dimensionality. *cedar* provides a graphical interface to build DFT models from a set of connectable core components. Models can be simulated in real time by numerically approximating the underlying continuous dynamical systems.

2.1 Dynamics and Processing Steps

An architecture in *cedar* is composed of *processing steps*. They update their *output* based on a number of current *inputs* that originate from other components of the architecture. Processing steps that implement dynamical systems are called *dynamics*. In addition to the inputs, their updates depend on their inner state. These inputs and outputs are formalized as sets of named *input* and *output slots*, whereas the inner states of dynamics use internal slots called *buffers*. These input and output slots are used to define connections from and to other components of the architecture.

Slots of *dynamics* contain *matrices* which represent the neural activation fields that are the inner state of the neural dynamics. In general, however, there are no restrictions on the type of data stored in slots. This enables us to work within the same framework when using neural dynamics as when we connect these dynamics to components that are based on non-neural data, for instance, at the interface to hardware devices.

Both *dynamics* and *processing steps* have adjustable parameters, which can be displayed, manipulated online, and stored as part of the overall architecture. This helps modelers to find suitable operational regimes for the dynamic neural fields.

2.2 Timing

In implementation on a computer-controlled robot or in computer simulation, the time courses of dynamical systems must be numerically approximated by iterative, discrete time steps. In *cedar*, this approximation is based on a solver of differential equations, using the forward Euler method. Although numerically not very efficient, the forward Euler lends itself to working with stochastic dynamics as well as to real-time implementation [7]. Individual components of the dynamics implement their update rule (i.e., the dynamics equation) based on their current state. A special element, the *looped trigger* (see Fig. 1), is responsible for periodically invoking this update with one of the following user-selectable timing schemes.

Fixed time step: The user specifies a fixed duration for the time step, which determines the minimum interval between iterations. If the duration of computation in an iteration step exceeds this fixed time step, the trigger skips an appropriate number of iterations or adapts its step size to the new duration.

Real time: Iterations are performed as often as possible and the time between two consecutive iterations is used as the time step. This means that the time step is determined by the amount of time it takes to update the overall architecture.

Simulated time: The trigger sends a fixed time step, regardless of how much time passes between iterations. This mode can be used to perform simulations that run faster than real time or to achieve numerical stability when other modes fail to do so.

A separate, non-looped triggering mechanism invokes updates of non-dynamic operations along the outgoing connections of dynamic components, thus creating an ordered chain of updates.

2.3 Connections

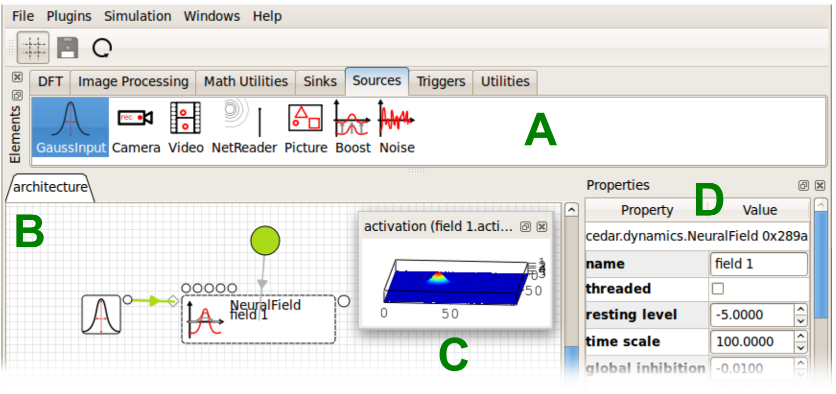
When different components of an architecture are connected by the user, the underlying framework checks the validity of these connections. They are *valid* if the type and properties of the connected output match the expectations of the receiving component. Potential semantic flaws result in a *warning*. Connections are *invalid* if they cannot be handled by the receiving component.

2.4 Graphical Notation and User Interface

Fig. 1 illustrates how two components, each of which implementing a DNF, are represented graphically. The small circles and diamonds attached to each component represent the input (left), buffer (top), and output (right) slots. Outputs from multiple other components may be connected to the diamond (for DNFs, these inputs are summed up). Input slots drawn in light-gray indicate that the components can be updated even when no input is connected. The green line between the neural fields represents a valid connection of the output of one field to the input of the other field (yellow lines would indicate warnings, red lines errors). The large round circle represents a looped trigger; the light gray lines originating from it indicate which updates it invokes.

cedar provides a graphical application (see Fig. 2) that offers tools for designing architectures in a drag-and-drop manner using this graphical notation.

The ability to plot the data in the slots of components allows for online inspection. A suitable plot class is chosen automatically, but users may also choose



Property	Value
cedar.dynamics.NeuralField 0x289a	
name	field 1
threaded	☐
resting level	-5.0000
time scale	100.0000
global inhibition	-0.0100

Fig. 2. *Components of the graphical user interface.* The pool of all available components is placed on top (A). It contains components available in *cedar* and the loaded plugins, grouped by theme. Desired components can be dragged onto the architecture canvas (B). Two display modes are demonstrated here: icon-only (left component) and icon-and-text ("field 1"). The current state of the neural field—the activation—is plotted in a separate window (C). The parameters of the field can be inspected and altered in the property pane (D).

other plot classes manually, e.g., for plotting matrices either as images or surface plots. New plot classes can be loaded at runtime via a plug-in structure.

Users may also inspect other properties of a component, for instance, the dimensionality and size of matrices. Online manipulation of parameters is accessible in the user interface as well.

The user interface provides the designer with feedback. Faulty connections and components that are in an erroneous state are highlighted in different colors. Additional features make the process of designing architectures more comfortable. When connecting components, for instance, all available end-points are highlighted and the validity of the potential connections is shown.

2.5 The DFT Toolbox

cedar provides essential components for DFT architectures, ordered into thematic groups.

The group *'DFT'* includes the core building blocks for architectures. The 'neural field' component provides a single-layer neural field of arbitrary dimensionality with lateral interactions [1]. 'Preshape' implements a memory trace for neural layers, which adapts to a given input over time, taking into account time scales for build-up and decay. Two processing steps, 'rate-to-space-code' and 'space-to-rate-code', transform population-based neural dynamics into rate-coded neural activity [14] and vice versa. These mechanisms are used to connect to the motor surface and proprioception. The processing step 'rate-matrix-to-space-code' interprets a matrix of rate-coded neurons along a space code metric.

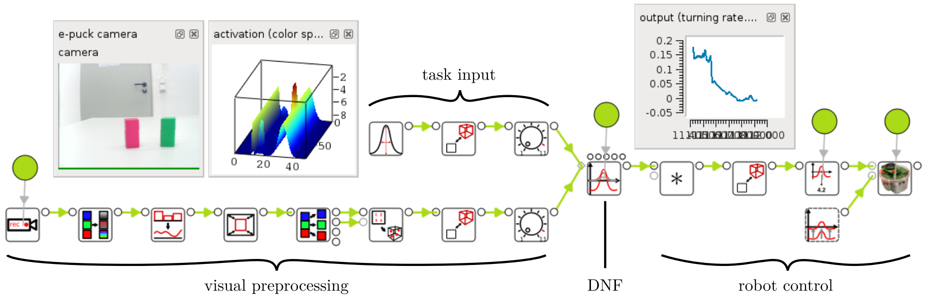


Fig. 3. *Screenshot of an exemplary architecture created using the graphical user interface of cedar* (groupings, e.g., 'task input', have been added manually). This DFT-based architecture controls an e-puck robot equipped with a color camera and makes it drive toward objects of specific colors, in this case the red block. The selection decision regarding the target is stabilized over time by a dynamic neural field.

All components in the group *sources* feed sensory input into architectures. *cedar* currently supports camera devices, video files, and images. Moreover, *cedar* includes artificially generated inputs, e.g., a homogeneous 'boost', a localized 'Gaussian input', and 'noise'. While noise is inherent in physical sensory inputs to an architecture, it is also an important tool during the tuning of an architecture, e.g., to test the stability of a system for various levels of noise. The source 'net reader' provides matrices that are sent by another process using a network-transparent protocol. The complementary component, 'net writer', is found in the group *sinks*. The current implementation uses YARP [8] to transfer these matrices between processes and workstations.

The group *'utilities'* offers solutions for connecting components of different dimensionality and granularity [14], for configuring the connection strength via scalar multiplication or arbitrary convolution of outputs, and general mathematical operations such as 'sum', 'component-wise multiplication', and 'coordinate transformation'.

cedar provides a group *'image processing'* with basic operations for closing the gap between image sources and population dynamics. It comprises a channel split and a color space conversion. More image processing operations (and other special-interest components) have been implemented in *plugins* which can be loaded at runtime. This keeps the core of *cedar* focused on modeling neural architectures with DFT.

3 Case Study

As an illustration, we implement a cognitive architecture in *cedar* and connect it to an e-puck, a small robot equipped with a differential drive and a color camera. The architecture is based on a model that selects a target by a given color cue (see Fig. 3). This paradigm is derived from a more complex phonotaxis robot [3].

The core element of the model is its decision mechanism, a dynamic neural field (DNF). It receives input from the visual sensory system about the location of salient target objects. Additionally, it receives an input that biases its decision toward a target of a specific color. The DNF selects one of the targets, stabilizes that decision against sensory fluctuations, and tracks the location of the target over time. The generated motor output orients the robot so that the selected object is centered in the camera image.

We implemented this model in the graphical user interface of *cedar*, composing the architecture from available elements by drag-and-drop (see Fig. 3) and tuning all parameters online. The resulting architecture consists of four parts: first, the visual preprocessing of the camera image; second, the task input that selects the range of colors the robot is attracted to; third, the DNF, which receives input from both the preprocessing and the task input; and fourth, the connection from the DNF to the robotic hardware. Please note that the architecture works in a closed loop through the environment in which the robot is embedded, even though this is not directly apparent from the figure. The sensory data impacts on the motor commands of the robot and vice versa.

The *visual preprocessing* consists of a sequence of image processing steps. They convert the camera's RGB-color images to saliency-based activation values which are then input to the DNF. The activation is defined over a two-dimensional space spanned by the horizontal viewing angle of the robot and a color metric. High activation values thus represent an object of a certain color at a certain horizontal location. The *task input* enters the DNF as a ridge of activation along the horizontal viewing angle (see, e.g., [14]) and biases the selection of objects toward a particular color. Once the DNF has formed a peak and thereby selected a target object, that peak *controls the robot.* The position of the peak along the horizontal axis is translated into rate-coded neural activation that indicates the direction of the colored object relative to the robot's current orientation. It is directly used as a turning rate of the robot, essentially turning the robot toward the object [3]. By adding a constant forward speed, the robot is able to drive toward the object. When presented with several targets, the robot selects the most salient one of the specified color and successfully drives toward it.

4 Conclusion

In the case study (Section 3), we illustrated *cedar* with an exemplary architecture derived from a simple cognitive model. The implementation of this model demonstrated how *cedar* addresses the key requirements on software that we identified in Section 1. *Ease of implementation* is achieved by using a graphical drag-and-drop interface for the assembly of architectures. This interface can also be used when *connecting physical devices* such as robots, to cognitive architectures. Architectures can be simulated, inspected, and parameterized in *real time*, enabling us to quickly assess the interdependence of regimes of different DNFs and the embodiment of cognitive architectures.

Adding new hardware devices is currently done individually for each new device. In our next major release, we plan to offer a more principled approach that reduces this overhead.

We also aim to further enhance the ease-of-use and scalability in future versions. *Groups*, which enable users to combine a set of components into reusable modules are planned as part of this effort, as are global parameters which can change multiple parameters simultaneously according to user-defined relations.

References

1. Amari, S.-I.: Dynamics of pattern formation in lateral-inhibition type neural fields. Biological Cybernetics 27, 77–87 (1977)
2. Bernardet, U., Verschure, P.F.M.J.: iqr: A tool for the construction of multi-level simulations of brain and behaviour. Neuroinformatics 8(2), 113–134 (2010)
3. Bicho, E., Mallet, P., Schöner, G.: Target representation on an autonomous vehicle with low-level sensors. International Journal of Robotics Research 19(5), 424–447 (2000)
4. Clark, A.: An embodied cognitive science? Trends in Cognitive Sciences 3(9), 345–351 (1999)
5. Eliasmith, C., Anderson, C.H.: Neural engineering: Computation, representation, and dynamics in neurobiological systems. MIT Press (2004)
6. Erlhagen, W., Schöner, G.: Dynamic Field Theory of movement preparation. Psychological Review 109(3), 545–572 (2002)
7. Kloeden, P.E., Platen, E.: Numerical solution of stochastic differential equations, 2nd edn. Springer (1999)
8. Metta, G., Fitzpatrick, P., Natale, L.: YARP: Yet another robot platform. International Journal on Advanced Robotics Systems 3(1), 43–48 (2006)
9. Richter, M., Sandamirskaya, Y., Schöner, G.: A robotic architecture for action selection and behavioral organization inspired by human cognition. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2457–2464. IEEE Press (2012)
10. Rumelhart, D.E., McClelland, J.L.: Parallel Distributed Processing: Explorations in the microstructure of cognition. Foundations, vol. 1. MIT Press, Cambridge (1986)
11. Schöner, G.: Dynamical systems approaches to cognition. In: Cambridge Handbook of Computational Cognitive Modeling, pp. 101–126. Cambridge University Press (2008)
12. Schutte, A.R., Spencer, J.P., Schöner, G.: Testing the Dynamic Field Theory: Working memory for locations becomes more spatially precise over development. Child Development 74, 1393–1417 (2003)
13. Stewart, T.C., Tripp, B., Eliasmith, C.: Python scripting in the Nengo simulator. Frontiers in Neuroinformatics 3 (2009)
14. Zibner, S.U., Faubel, C., Iossifidis, I., Schöner, G.: Dynamic Neural Fields as building blocks for a cortex-inspired architecture of robotic scene representation. IEEE Transactions on Autonomous Mental Development 3(1) (2011)

Adaptive Critic Neural Network Solution of Optimal Control Problems with Discrete Time Delays

Tibor Kmet[1] and Maria Kmetova[2]

[1] Constantine the Philosopher University, Department of Informatics,
Tr. A. Hlinku 1, 949 74 Nitra, Slovakia
tkmet@ukf.sk
http://www.ukf.sk

[2] Constantine the Philosopher University, Department of Mathematics,
Tr. A. Hlinku 1, 949 74 Nitra, Slovakia
mkmetova@ukf.sk

Abstract. A neural network based optimal control synthesis is presented for solving optimal control problems with discrete time delays in state and control variables subject to a control and state constraints. The optimal control problem is transcribed into nonlinear programming problem which is implemented with feed forward adaptive critic neural network to find optimal control and optimal trajectory. The proposed simulation methods is illustrated by the optimal control problem of nitrogen transformation cycle model with discrete time delay of nutrient uptake. Results show that adaptive critic based systematic approach are promising in obtaining the optimal control with discrete time delays in state and control variables subject to control and state constraints.

Keywords: optimal control problem with delays, state and control constraints, recurrent neural network, adaptive critic synthesis, numerical examples, nitrogen transformation cycle.

1 Introduction

Optimal control of nonlinear systems with discrete time delays in state and control variables is one of the most active subjects in control theory. There are rarely analytical solutions [4] although several numerical computation approaches have been proposed e.g. see [6], [7], [13], [17]. The most of the literature dealing with numerical methods for the solution of general optimal control problems focuses on algorithms for solving discretized problems. The basic idea of these methods is to apply nonlinear programming techniques to the resulting finite dimensional optimization problem [2], [6]. Then neural networks are used as universal function approximation to solve finite dimensional optimization problems forward in time with "adaptive critic designs" [11], [12], [18]. For the neural network, a feed forward neural network with one hidden layer, a steepest descent error backpropagation rule, a hyperbolic tangent sigmoid transfer function and a linear transfer function were used.

© Springer-Verlag Berlin Heidelberg 2013

The paper presented extends adaptive critic neural network architecture proposed by [9] to the optimal control problems with discrete time delays in state and control variables subject to control and state constraints. This paper is organized as follows. In Section 2, optimal control problems with delays in state and control variables subject to control and state constraints are introduced. We summarize the necessary optimality conditions, give a short overview of the basic results including the iterative numerical methods. In Section 3, we discuss the discretization methods for the given optimal control problem and formulate the resulting nonlinear programming problems. Section 4 presents a short description of adaptive critic neural network synthesis for the optimal control problem with delays in state and control variables subject to control and state constraints. We also present a new algorithm to solve optimal control problems. In Section 5, we present a description of the model of nitrogen transformation cycle with discrete time delay in nutrients uptake. We apply the new proposed methods to the model presented to compare short-term and long-term strategies of nutrients uptake by phytoplankton. Numerical results are also given. Conclusions are being presented in Section 6.

2 The Optimal Control Problem

We consider the nonlinear control problem with delays in state and control variables subject to control and state constraints. Let $x(t) \in R^n$ and $u(t) \in R^m$ denote the state and control variable, respectively in a given time interval $[t_0, t_f]$. The optimal control problem is to minimize

$$\mathcal{J}(u) = g(x(t_f)) + \int_{t_0}^{t_f} f_0(x(t), x(t-\tau_x), u(t), u(t-\tau_u))dt \tag{1}$$

subject to

$$\begin{aligned} \dot{x}(t) &= f(x(t), x(t-\tau_x), u(t), u(t-\tau_u)), \\ x(t) &= \phi_s(t), \quad u(t) = \phi_c(t), \quad t \in [t_0 - \tau_u, t_0], \\ \psi(x(t_f)) &= 0, \quad c(x(t), u(t)) \le 0, \quad t \in [t_0, t_f], \end{aligned}$$

where $\tau_x \ge 0$ and $\tau_u \ge 0$ are discrete time delay in the state and control variable, respectively. The functions $g : R^n \to R$, $f_0 : R^{n+m} \to R$, $f : R^{n+m} \to R^n$, $c : R^{n+m} \to R^q$ and $\psi : R^{n+m} \to R^r$, $0 \le r \le n$ are assumed to be sufficiently smooth on appropriate open sets and the initial conditions $\phi_s(t)$, $\phi_c(t)$ are continuous functions. The theory of necessary conditions for the optimal control problem of form (1) is well developed, see e.g. [6], [7], [14]. We introduce an additional state variable

$$x_0(t) = \int_0^t f_0(x(s), x(s-\tau_x), u(s), u(s-\tau_u)ds$$

defined by the

$$\dot{x}_0(t) = f_0(x(t), x(t-\tau_x), u(t), u(t-\tau_u), \; x_0(t) = 0, \; t \in [t_0 - \tau_x, t_0].$$

Then the augmented Hamiltonian function for problem (1) is

$$\mathcal{H}(x, x_{\tau_x}, u, u_{\tau_u}, \lambda, \mu) = \sum_{j=0}^{n} \lambda_j f_j(x, x_{\tau_x}, u, u_{\tau_u}) + \sum_{j=0}^{q} \mu_j c_j(x, u),$$

where $\lambda \in R^{n+1}$ is the adjoint variable and $\mu \in R^q$ is a multiplier associated to the inequality constraints. Assume that $\tau_x,\ \tau_u \geq 0,\ (\tau_x, \tau_u) \neq (0,0)$ and $\frac{\tau_x}{\tau_u} \in \mathbb{Q}$ for $\tau_u > 0$ or $\frac{\tau_u}{\tau_x} \in \mathbb{Q}$ for $\tau_x > 0$. Let $(\hat{x}, \hat{u})$ be an optimal solution for (1.) Then the necessary optimality condition for (1) implies [6] that there exist a piecewise continuous and piecewise continuously differentiable adjoint function $\lambda : [t_0, t_f] \to R^{n+1}$, a piecewise continuous multiplier function $\mu : [t_0, t_f] \to R^q,\ \hat{\mu}(t) \geq 0$ and a multiplier $\sigma \in R^r$ satisfying

$$\begin{aligned}
\dot{\lambda}_j(t) &= -\frac{\partial \mathcal{H}}{\partial x_j}\left(\hat{x}(t), \hat{x}(t-\tau_x), \hat{u}(t), \hat{u}(t-\tau_u), \lambda(t), \mu(t)\right) - \\
&\quad \chi_{[t_0, t_f - \tau_x]} \frac{\partial \mathcal{H}}{\partial x_{\tau_x j}}(\hat{x}(t+\tau_x), \hat{x}(t), \hat{u}(t+\tau_x), \hat{u}(t-\tau_u+\tau_x), \lambda(t+\tau_x), \mu(t+\tau_x)), && (2)\\
\lambda_j(t_f) &= g_{x_j}(\hat{x}(t_f)) + \sigma \psi_{x_j}(\hat{x}(t_f)),\ j = 0, \ldots, n, && (3)\\
0 &= -\frac{\partial \mathcal{H}}{\partial u_j}\left(\hat{x}(t), \hat{x}(t-\tau_x), \hat{u}(t), \hat{u}(t-\tau_u), \lambda(t), \mu(t)\right) - \\
&\quad \chi_{[t_0, t_f - \tau_u]} \frac{\partial \mathcal{H}}{\partial u_{\tau_u j}}\left(\hat{x}(t+\tau_u), \hat{x}(t-\tau_x+\tau_u), \hat{u}(t+\tau_u), \hat{u}(t), \lambda(t+\tau_u), \mu(t+\tau_u)\right), && (4)\\
&\quad j = 1, \ldots, m.
\end{aligned}$$

Furthermore, the complementary conditions hold, i.e. in $t \in [t_0, t_f],\ \mu(t) \geq 0$, $c(x(t), u(t)) \leq 0$ and $\mu(t)c(x(t), u(t)) = 0$. Herein, the subscript $x,\ x_{\tau_x},\ u$ and u_{τ_u} denotes the partial derivative with respect to $x,\ x_{\tau_x},\ u$ and u_{τ_u}, respectively.

3 Discretization of the Optimal Control Problem

The direct optimization methods for solving the optimal control problem are based on a suitable discretization of (1), see e.g. [2], [6]. We assume that $\tau_u = l\frac{\tau_x}{k}$ with $l, k \in \mathbb{N}$. Defining $h_{max} = \frac{\tau_x}{k}$ gives the maximum interval length for an elementary transformation interval that satisfies $\frac{\tau_x}{h_{max}} = k \in \mathbb{N}$ and $\frac{\tau_u}{h_{max}} = l \in \mathbb{N}$. The minimum grid point number for an equidistant discretization mesh $N_{min} = \frac{t_f - t_0}{h_{max}}$. Choose a natural number $K \in \mathbb{N}$ and set $N = KN_{min}$. Let $t_i \in \langle t_0, t_f \rangle,\ i = 0, \ldots, N$, be an equidistant mesh point with $t_i = t_0 + ih, i = 0, \ldots, N$, where $h = \frac{b-a}{N}$ is a time step and $t_f = Nh + t_0$. Let the vectors $x^i \in R^{n+1},\ u^i \in R^m, i = 0, \ldots, N$, be an approximation of the state variable and control variable $x(t_i),\ u(t_i)$, respectively at the mesh point t_i. Euler's approximation applied to the differential equations yields $x^{i+1} = x^i + hf(x^i, x^{i-k}, u^i, u^{i-l}),\ i = 0, \ldots, N-1$. Choosing the optimal variable $z := (x^0, x^1, \ldots, x^{N-1}, u^0, \ldots, u^{N-1}) \in R^{N_s},\ N_s = (n+m)N,$ the optimal control problem is replaced by the following discretized control problem in the form of nonlinear programming problem with inequality constraints: Minimize

$$\mathcal{J}(z) = \mathcal{G}(x^N) = g((x_1, \ldots, x_n)^N) + x_0^N \qquad (5)$$

subject to

$$\begin{aligned}
x^{i+1} &= x^i + hf(x^i, x^{i-k}, u^i, u^{i-l}),\ i = 0, \ldots, N-1, \qquad (6)\\
x^{-i} &= \phi_x(t_0 - ih),\quad i = k, \ldots, 0,\quad u^{-i} = \phi_u(t_0 - ih),\quad i = l, \ldots, 0,\\
\psi(x^N) &= 0,\quad c(x^i, u^i) \le 0,\ i = 0, \ldots, N-1.
\end{aligned}$$

In a discrete-time formulation we want to find an admissible control which minimizes objective function (5). Let us introduce the Lagrangian function for the nonlinear optimization problem (5):

$$\begin{aligned}
\mathcal{L}(z, \lambda, \sigma, \mu) = &\sum_{i=0}^{N-1} \lambda^{i+1}(-x^{i+1} + x^i + hf(x^i, x^{i-k}, u^i, u^{i-l})) + \mathcal{G}(x^N) + \\
&\sum_{i=0}^{N-1} \mu^i c(x^i, u^i) + \sigma\psi(x^N).
\end{aligned}$$

The first order optimality conditions of Karush-Kuhn-Tucker [13] for the problem (5) are:

$$\begin{aligned}
0 \ = \ \mathcal{L}_{x^i}(z, \lambda, \sigma, \mu) &= \lambda^{i+1} - \lambda^i + h\lambda^{i+1} f_{x^i}(x^i, x^{i-k}, u^i, u^{i-l}) + \qquad (7)\\
&\quad h\lambda^{i+k+1} f_{x^i_{\tau_x}}(x^{i+k}, x^i, u^{i+k}, u^{i-l+k}) + \mu^i c_{x^i}(x^i, u^i),\\
&\quad i = 0, \ldots, N-k-1,\\
0 \ = \ \mathcal{L}_{x^i}(z, \lambda, \sigma, \mu) &= \lambda^{i+1} - \lambda^i + h\lambda^{i+1} f_{x^i}(x^i, x^{i-k}, u^i, u^{i-l}) + \mu^i c_{x^i}(x^i, u^i),\\
&\quad i = N-k, \ldots, N-1,\\
0 \ = \ \mathcal{L}_{x^N}(z, \lambda, \sigma, \mu) &= \mathcal{G}_{x^N}(x^N) + \sigma\psi_{x_N}(x_N) - \lambda^N, \qquad (8)\\
0 \ = \ \mathcal{L}_{u^i}(z, \lambda, \sigma, \mu) &= h\lambda^{i+1} f_{u^i}(x^i, x^{i-k}, u^i, u^{i-l}) +\\
&\quad h\lambda^{i+l+1} f_{u^i_{\tau_u}}(x^{i+l}, x^{i-k+l}, u^{i+l}, u^i) + \mu^i c_{u^i}(x^i, u^i),\\
&\quad i = 0, \ldots, N-l-1, \qquad (9)\\
0 \ = \ \mathcal{L}_{u^i}(z, \lambda, \sigma, \mu) &= h\lambda^{i+1} f_{u^i}(x^i, x^{i-k}, u^i, u^{i-l}) + \mu^i c_{u^i}(x^i, u^i),\\
&\quad i = N-l, \ldots, N-1.
\end{aligned}$$

Eqs. (7) – (10) represent the discrete version of the necessary condition (2) - (4) for optimal control problem (1).

4 Adaptive Critic Neural Network for an Optimal Control Problem with Control and State Constraints

It is well known that a neural network can be used to approximate the smooth time-invariant functions and the uniformly time-varying functions [3], [16]. Experience has shown that optimization of functionals over admissible sets of functions made up of linear combinations of relatively few basis functions with a simple structure and depending nonlinearly on a set of "inner" parameters e.g., feedforward neural networks with one hidden layer and linear output activation

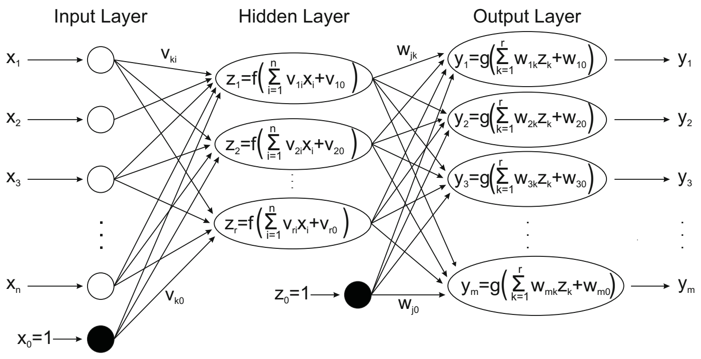


Fig. 1. Feed forward neural network topology with one hidden layer, v_{ki}, w_{jk} are values of connection weights, v_{k0}, w_{j0} are values of bias, f(.), g(.) are activation functions

units often provides surprisingly good suboptimal solutions [1], [5], [10]. Fig. 1 shows a feed forward neural network with n inputs node, one hidden layer of r units and m output units. Let $x = [x_1, \ldots, x_n]'$ and $y = [y_1, \ldots, y_m]'$ be the input and output vectors of the network, respectively. Let $V = [v_1, \ldots, v_r]'$ be the matrix of synaptic weights between the input nodes and the hidden units, where $v_k = [v_{k0}, v_{k1} \ldots, v_{kn}]$; v_{k0} is the bias of the kth hidden unit, and v_{ki} is the weight that connects the ith input node to the kth hidden unit.

Let also $W = [w_1, \ldots, w_m]'$ be the matrix of synaptic weights between the hidden and output units, where $w_j = [w_{j0}, w_{j1} \ldots, w_{jr}]$; w_{j0} is the bias of the jth output unit, and w_{jk} is the weight that connects the kth hidden unit to the jth output unit. The response of the kth hidden unit is given by $z_k = tanh\left(\sum_{i=0}^{n} v_{ki}x_i\right)$, $k = 1, \ldots, r$, where tanh(.) is the activation function for the hidden units. The response of the jth output unit is given by $y_j = \sum_{k=0}^{r} w_{jk}z_k$, $j = 1, \ldots, m$. The multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between the input and output vectors. The number of neurons in the input and output layers is given by the number of input and output variables, respectively. The multi-layered feed forward network shown in Fig. 2 is trained using the steepest descent error backpropagation rule. Basically, it is a gradient descent, a parallel distributed optimization technique to minimize the error between the network and the target output [15]. To solve the equations (9) we are concerned with the following nonlinear projection equation (for detail description see [19]):

$$\alpha\mathbb{F}(\mathbb{P}_X(y)) + y - \mathbb{P}_X(y) = 0, \tag{10}$$

where $\alpha > 0$ is a constant, $\mathbb{F} : R^l \to R^l$, $X = \{y \in R^l \mid y_{imin} \leq y_i \leq y_{imax}\}$ and $\mathbb{P}_X : R^l \to X$ is a projection operator defined by $\mathbb{P}_X(y) = (\mathbb{P}_X(y_1), \ldots, \mathbb{P}_X(y_l))$

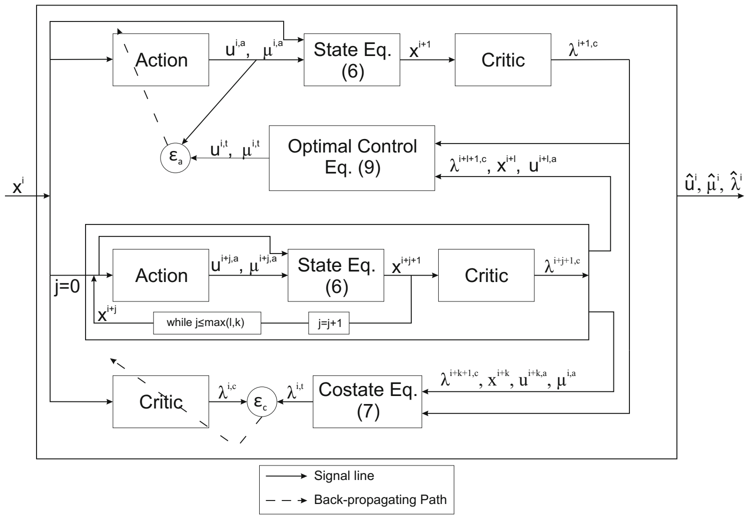


Fig. 2. Architecture of adaptive critic feed forward network synthesis, x^i-input signal to the action and critic network, $\hat{u}^{i,a}, \hat{\mu}^{i,a}$ and $\hat{\lambda}^{i,c}$ are output signal from action and critic network, respectively and $\hat{u}^{i,t}, \hat{\mu}^{i,t}$ and $\hat{\lambda}^{i,t}$ are solutions of equation (9) and co-state equation (7), respectively

$$\mathbb{P}_X(y_i) = \begin{cases} y_{imin} & : \quad y_i < y_{imin} \\ y_i & : \quad y_{imax} \le y_i \le y_{imax} \\ y_{imax} & : \quad y_i > y_{imax}, \end{cases}$$

which can be solved by the following dynamic model

$$\dot{y}(t) = -\beta(\alpha \mathbb{F}(\mathbb{P}_X(y)) + y - \mathbb{P}_X(y)). \tag{11}$$

Note that y_{imin} and y_{imax} are lower and upper limits of y_i, $i = 1, \ldots, l$. Asymptotic and exponential stability of the present recurrent neural network (11) are proven in [19]. The equilibrium points of (11) coincide with the solutions of (10). We can state the algorithm to solve the optimal control problem using the adaptive critic and recurrent neural network. In the Pontryagin's maximum principle for deriving an optimal control law, the interdependence of the state, costate and control dynamics is made clear. Indeed, the optimal control $\hat{u}$ and multiplier $\hat{\mu}$ is given by Eq. (9), while the costate Eqs. (7) - (8) evolves backward in time and depends on the state and control. The adaptive critic neural network [12] is based on this relationship is shown in Fig. 2. It consists of two networks at each node: an action network, the inputs for which are the current states and its outputs are the corresponding control $\hat{u}$ and multiplier $\hat{\mu}$, and the critic network for which the current states are inputs and current costates are outputs for normalizing the inputs and targets (zero mean and standard deviations). For detail explanation see [15]. Based on Fig. 2 the adaptive critic neural network procedure of the optimal control problem is summarized in Algorithm 1.

Algorithm 1. Algorithm to solve the optimal control problem.

```
Input: Choose $t_0$, $t_f$, $N$ - number of steps, time step $h$, $\alpha > 0$, $\beta > 0$, $\varepsilon_a$, $\varepsilon_c$
       and $\varepsilon_{rnn}$ - stopping tolerance for action, critic and recurrent neural
       network, respectively, $x^{-i} = \phi_s(t_0 - ih),\ i = k, \ldots, 0$,
       $u^{-i} = \phi_c(t_0 - ih),\ i = l, \ldots, 0$ -initial values.
Output: Set of final approximate optimal control $\hat{u}(t_0 + ih) = \hat{u}^i$ and optimal
        trajectory $\hat{x}(t_0 + (i+1)h) = \hat{x}^{i+1},\ i = 0, \ldots, N-1$, respectively
1  Set the initial weight $\mathbb{W}^a = (V^a, W^a),\ \mathbb{W}^c = (V^c, W^c)$
   for $i \leftarrow 0$ to $N-1$ do
2      while $err_a \geq \epsilon_a$ and $err_c \geq \epsilon_c$ do
3          for $j \leftarrow 0$ to $max(k,l)$ do
4              Compute $u^{i+j,a}$, $\mu^{i+j,a}$ and $\lambda^{i+j+1,c}$ using action ($\mathbb{W}^a$) and critic
               ($\mathbb{W}^c$) neural networks, respectively and $x^{i+j+1}$ by Eq. (6)
5          Compute $\lambda^{i,t}$, $u^{i,t}$, and $\mu^{i,t}$ using Eqs. (7), (9) and (11) with
           $X = \{(u^i, \mu^i) \in R^{m+q} | \mu^i \geq 0\}$,
           $\mathbb{F}(u^i, \mu^i) = (\mathcal{L}_{u^i}(z, \lambda, \sigma, \mu), -c(x^i, u^i))$ and stopping tolerance $\epsilon_{rnn}$.
6          if $i = N-1$ then
7              $X = \{(u^{N-1}, \mu^{N-1}, \sigma) \in R^{m+q+r} | \mu^{N-1} \geq 0,\}$,
               $\mathbb{F}(u^{N-1}, \mu^{N-1}, \sigma) = (\mathcal{L}_{u^{N-1}}(z, \lambda, \sigma, \mu), -c(x^{N-1}, u^{N-1}), -\psi(x^N))$
               with $\lambda^N = \mathcal{G}_{x^N}(x^N) + \sigma\psi_{x_N}(x_N)$
8          $err_c = \| \lambda^{i,t} - \lambda^{i,c} \|$
9          $err_a = \| (u,\mu)^{i,t} - (u,\mu)^{i,a} \|$
10         With the data set $x^i$, $\lambda^{i,t}$ update the weight parameters $\mathbb{W}^c$
11         With the data set $x^i$, $(u,\mu)^{i,t}$ update the weight parameters $\mathbb{W}^a$
12         Set $\lambda^{i,c} = \lambda^{i,t}$, $(u,\mu)^{i,a} = (u,\mu)^{i,t}$
13     Set $\hat{\lambda}^i = \lambda^{i,t}$, $(\hat{u}^i, \hat{\mu}^i) = (u,\mu)^{i,t}$
14     Compute $\hat{x}^{i+1}$ using Eq. (6) and $\hat{u}^i$
15     return $\hat{\lambda}^i$, $\hat{u}^i, \hat{\mu}^i$, $\hat{x}^{i+1}$
```

In the adaptive critic synthesis, the action and critic network were selected such that they consist of $n+m$ subnetworks, respectively, each having $n-3n-1$ structure (i.e. n neurons in the input layer, $3n$ neurons in the hidden layer and one neuron in the output layer). The training procedure for the action and critic networks, respectively are given by [12]. From the free terminal condition ($\psi(x) \equiv 0$) from Eqs. (7) - (8) we obtain that $\lambda_0^i = -1$, $i = N, \ldots, 0$ and $\lambda_j^N = 0$, $j = 1, \ldots, N$. We use this observation before proceeding to the actual training of the adaptive critic neural network. Further discussion and detail explanation of these adaptive critic methods can be found in [9], [11], [12] and [15].

5 Nitrogen Transformation Cycle

The aerobic transformation of nitrogen compounds [8] includes: Specific groups of microorganisms participate in transformation of nitrogen compounds.

Heterotrophic bacteria (x_1) assimilate and decompose the soluble organic nitrogen compounds DON (x_6) derived from detritus (x_5). Ammonium (x_7), one of the final decomposition products undergoes a biological transformation into nitrate (x_9). This is carried out by aerobic chemoautotrophic bacteria in two stages: ammonia is first oxidized by nitrifying bacteria from the genus Nitrosomonas (x_2) into nitrites (x_8) that serve as an energy source for nitrating bacteria mainly from the genus Nitrobacter (x_3). The resulting nitrates may be assimilated together with ammonia and soluble organic forms of nitrogen by the phytoplankton (x_4), whereby the aerobic transformation cycle of nitrogen compounds is formed. The individual variables $x_1, \ldots, x_9$ represent nitrogen concentrations contained in the organic as well as in inorganic substances and living organisms presented in a model. The following system of ordinary differential equations is proposed as a model for the nitrogen transformation cycle:

$$\begin{aligned}
\dot{x}_i(t) &= x_i(t)U_i(x(t)) - x_i(t)E_i(x(t)) - x_i(t)M_i(x(t))),\ i = 1,2,3,\\
\dot{x}_4(t) &= x_4(t-\tau)(U_4(x(t-\tau)) - E_i(x(t-\tau)) - M_i(x(t-\tau))),\\
\dot{x}_5(t) &= \sum_{i=1}^{4} x_i M_i(x) - K_5 x_5(t),\\
\dot{x}_6(t) &= K_5 x_5(t) - x_1(t)U_1(x(t)) + x_4(t)E_4(x(t)) - x_4(t)P_6(x(t)),\\
\dot{x}_7(t) &= x_1(t)E_1(x(t)) - x_2(t)U_2(x(t)) - x_4(t)P_7(x(t)),\\
\dot{x}_8(t) &= x_2(t)E_2(x(t)) - x_3(t)U_3(x(t)),\\
\dot{x}_9(t) &= x_3(t)E_3(x(t)) - x_4(t)P_9(x(t)),
\end{aligned} \tag{12}$$

where $x_i(t)$ are the concentration of the recycling matter in microorganisms, the available nutrients and detritus, respectively. The constant τ stands for the discrete time delay in uptake of nutrients by phytoplankton. Functions occurring in the model are given in Table 1 in ecological and mathematical notation, respectively. Three variables $u = (u(1), u(2), u(3))$ express the preference coefficients for update of $x_6,\ x_7,\ x_9$. It can be expected that the phytoplankton will employ control mechanisms in such a way as to maximize its biomass over a given period t_f of time:

$$J(u) = \int_0^{t_f} x_4(t)dt \rightarrow max \tag{13}$$

under the constraint

$$\begin{aligned}
C(x,u) &:= b_1U_4(x,u) + b_2P_6(x,u) + b_3P_9(x,u) + b_4E_4(x,u) \le W(I), \\
&\quad u_i \in [0, u_{imax}] \ \ for\ i = 1,2,3.
\end{aligned} \tag{14}$$

The last inequality expresses the fact that amount of energy used for "living expenses" (synthesis, reduction and excretion of nutrients) by phytoplankton cannot exceed a certain value $W(I)$ which depends on light intensity I (for detail explanation see [8]). We are led to the following optimal control problems: (1) instantaneous maximal biomass production with respect to u:

Table 1. Description of functions occurring in the model

$U_i(x)$	$= \frac{K_i x_{i+5}}{1+g_i x_{i+5}},\ i=1,2,3$	
p	$= u_1x_6 + u_2x_7 + u_3x_9$	
$U_4(x)$	$= \frac{K_4 p}{1+g_4 p}$	U_i - uptake rate
$L_i(x)$	$= \frac{a_{2i-1}U_i(x)}{1+a_{2i}U_i(x)} + 1 - \frac{a_{2i-1}}{a_{2i}}$	L_i - excretion activity
$M_i(x)$	$= g_{2i+3} + g_{2i+4}L_i(x)$	M_i - mortality rate
$E_i(x)$	$= U_i(x)L_i(x),\ i=1,\ldots,4$	E_i - excretion rate
$P_i(x)$	$= \frac{K_4 u_i x_i}{1+g_4 p},\ \ i=6,7,9$	P_i - uptake rate.

$$\dot{x}_4 = x_4(U_4(x,u) - E_4(x,u) - M_4(x,u)) \to max \tag{15}$$

under the constraint $C(x,u) \leq W(I)$, for all $t \in [t_0, t_f]$ and $u_i \in [0, u_{imax}]$, i=1,2,3 (To maximize (15) is equivalent to find the maximum of the function

$$p(u) = u_1x_6 + u_2x_7 + u_3x_9$$

under the constraint $C(x,u) \leq W(I)$, $u_i \in [0, u_{imax}]$ for i=1,2,3.),

(2) global maximal biomass production with respect to u:

$$J(u) = \int_{t_0}^{t_f} x_4(t)dt \to max \tag{16}$$

under the constraint $C(x,u) \leq W(I)$, for all $t \in [t_0, t_f]$ and $u_i \in [u_{imin}, u_{imax}]$ for i=1,2,3. We introduce an additional state variable

$$x_0(t) = \int_0^t x_4(s)ds. \tag{17}$$

We are led to the following optimal control problem: Maximize

$$x_0(t_f) \tag{18}$$

under the constraints

$$\begin{aligned} c_1(x,u) &= C(x,u) - W(I), \leq 0 \\ c_{i+1}(x,u) &= u_{imin} - u_i \leq 0, \\ c_{i+4}(x,u) &= u_i - u_{imax} \leq 0,\ i=1,2,3. \end{aligned}$$

Discretization of Eqs. (12) and (13) using Eqs. (7) – (8) and (5) leads to minimize

$$-x_0^N$$

subject to

$$
\begin{aligned}
x^{i+1} &= x^i + hF(x^{i-k}, x^i, u^i, u^{i-l}),\ i = 0, \ldots, N-1, \\
\lambda^i &= \lambda^{i+1} + h\lambda^{i+1} F_{x^i}(x^{i-k}, x^i, u^i, u^{i-l}) + \\
&\quad h\lambda^{i+k+1} F_{x^i_{\tau_x}}(x^{i+k}, x^i, u^{i+k}, u^{i-l+k}) + \mu^i c_{x^i}(x^i, u^i), \qquad (19) \\
\lambda^i_0 &= -1,\ i = 0, \ldots, N-1, \\
\lambda^N &= (-1, 0, 0, 0, 0, 0, 0, 0, 0, 0), \qquad (20) \\
0 &= h\lambda^{i+1} F_{u^i}(x^{i-k}, x^i, u^i, u^{i-l}) + \\
&\quad h\lambda^{i+l+1} F_{u^i_{\tau_u}}(x^{i+l}, x^{i-k+l}, u^{i+l}, u^i) + \mu^i c_{u^i}(x^i, u^i),
\end{aligned}
$$

where the vector function $F(x, u) = (-x_4, F_1(x, u), \ldots, F_9(x, u))$ is given by Eq. (13) and by right-hand side of Eq. (12).

5.1 Numerical Simulation

The solution of optimal control problem (18) with state and control constraints using adaptive critic neural network and NLP methods are displayed in Fig. 3. In the adaptive critic synthesis, the critic and action network were selected such

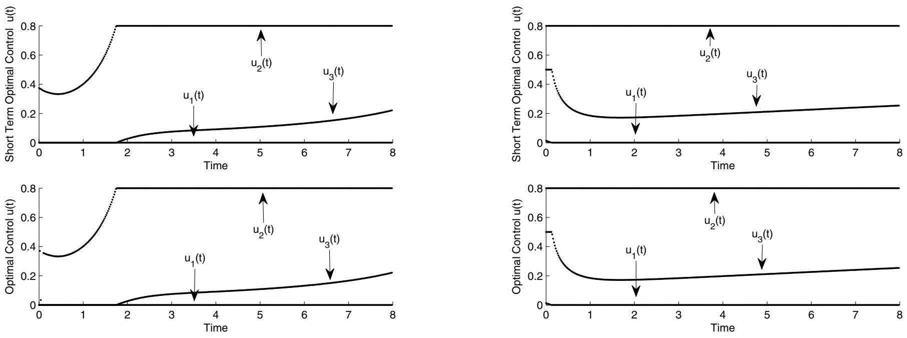


Fig. 3. Adaptive critic neural network simulation of optimal control $\hat{u}(t)$ and $\bar{u}(t)$ with initial condition $\psi_s(t) = (0.1, 0.1, 0.2, 0.8, 0.4, 0.5, 0.6, 0.7, .1)$ and $\psi_c(0) = (0.01, 0.01, 0.02, 0.08, 0.04, 0.05, 0.06, 0.07, .01)$, respectively for $t \in [-1, 0]$

that they consist of nine and four subnetworks, respectively, each having 9-27-1 structure (i.e. nine neurons in the input layer, twenty seven neurons in the hidden layer and one neuron in the output layer). The proposed neural network is able to meet the convergence tolerance values that we choose, which led to satisfactory simulation results. Simulations show that there is a very good agreement between short-term and long-term strategy and proposed neural network is able to solve nonlinear optimal control problem with state and control constraints. The optimal strategy is the following. In the presence of high ammonium concentration, the uptake of DON and nitrate is stopped. If the concentration of ammonium drops below a certain limit value, phytoplankton start to assimilate DON or nitrate dependently on values b_2, b_3. If the concentration of all three forms of nitrogen are low, all of them are assimilated by phytoplankton at the maximal possible rate, e.i. $\hat{u}_i(t) = u_{imax}$ for all $t \in [t_0, t_f]$ (Figure 3). Our results are quite similar to those obtained in [8] by using Pontriagin's maximum principle.

6 Conclusion

A single new network adaptive critic approach is presented for optimal control synthesis with discrete time delay in state and control variables subject to control and state constraints. Using Euler's methods the optimal control problem is transcribed into a discrete-time high-dimensional nonlinear programming problem. Adaptive critic neural network and the iterative programming algorithm were developed to seek for the state, costate and control variables of the constrained optimal control problem with time delay. These approach is applicable to wide class of nonlinear systems. Simulation studies have demonstrated with an optimal control problems related to nitrogen transformation cycle including phytoplankton production. Using MATLAB, a simple simulation model based on adaptive critic neural network was constructed. Numerical simulations have shown that adaptive critic neural network is able to solve nonlinear optimal control problem with discrete time delay and with control and state constraints.

Acknowledgment. The paper was worked out as a part of the solution of the scientific project number KEGA 004UJS-4/2011 and VEGA 1/0699/12.

References

1. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory 39, 930–945 (1993)
2. Buskens, C., Maurer, H.: SQP-methods for solving optimal control problems with control and state constraints: adjoint variable, sensitivity analysis and real-time control. Journal of Computational and Applied Mathematics 120, 85–108 (2000)
3. Hornik, M., Stichcombe, M., White, H.: Multilayer feed forward networks are universal approximators. Neural Networks 3, 256–366 (1989)
4. Hrinca, I.: An Optimal Control Problem for the Lotka-Volterra System with Delay. Nonlinear Analysis, Theory, Methods, Applications 28, 247–262 (1997)
5. Gnecco, A.: A Comparison Between Fixed-Basis and Variable-Basis Schemes for Function Approximation and Functional Optimization. Journal of Applied Mathematics 2012, article ID 806945 (2012)
6. Gollman, L., Kern, D., Mauer, H.: Optimal control problem with delays in state and control variables subject to mixed control-state constraints. Optim. Control Appl. Meth. 30, 341–365 (2009)
7. Kirk, D.E.: Optimal Control Theory: An Introduction. Dover Publications, Inc., Mineola (1989)

8. Kmet, T.: Material recycling in a closed aquatic ecosystem. I. Nitrogen transformation cycle and preferential utilization of ammonium to nitrate by phytoplankton as an optimal control problem. Bull. Math. Biol. 58, 957–982 (1996)
9. Kmet, T.: Neural network solution of optimal control problem with control and state constraints. In: Honkela, T. (ed.) ICANN 2011, Part II. LNCS, vol. 6792, pp. 261–268. Springer, Heidelberg (2011)
10. Makozov, Y.: Uniform approximation by neural networks. Journal of Approximation Theory 95, 215–228 (1998)
11. Padhi, R., Unnikrishnan, N., Wang, X., Balakrishnan, S.N.: Adaptive-critic based optimal control synthesis for distributed parameter systems. Automatica 37, 1223–1234 (2001)
12. Padhi, R., Balakrishnan, S.N., Randoltph, T.: A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems. Neural Networks 19, 1648–1660 (2006)
13. Polak, E.: Optimization Algorithms and Consistent Approximation. Springer, Heidelberg (1997)
14. Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V., Mischenko, E.F.: The Mathematical Theory of Optimal Process. Nauka, Moscow (1983) (in Russian)
15. Rumelhart, D.F., Hinton, G.E., Wiliams, R.J.: Learning internal representation by error propagation. In: Rumelhart, D.E., McClelland, D.E. (eds.) PDP Research Group: Parallel Distributed Processing: Foundation, vol. 1, pp. 318–362. The MIT Press, Cambridge (1987)
16. Sandberg, E.W.: Notes on uniform approximation of time-varying systems on finite time intervals. IEEE Transactions on Circuits and Systems-1: Fundamental Theory and Applications 45, 305–325 (1998)
17. Sun, D.Y., Huang, T.C.: A solutions of time-delayed optimal control problems by the use of modified line-up competition algorithm. Journal of the Taiwan Institute of Chemical Engineers 41, 54–64 (2010)
18. Werbos, P.J.: Approximate dynamic programming for real-time control and neural modelling. In: White, D.A., Sofge, D.A. (eds.) Handbook of Intelligent Control: Neural Fuzzy, and Adaptive Approaches, pp. 493–525 (1992)
19. Xia, Y., Feng, G.: A New Neural Network for Solving Nonlinear Projection Equations. Neural Network 20, 577–589 (2007)

Emotion Generation System Considering Complex Emotion Based on MaC Model with Neural Networks

Tsubasa Takamatsu and Yuko Osana

Tokyo University of Technology,
1404-1 Katakura Hachioji, Tokyo, Japan
osana@stf.teu.ac.jp

Abstract. In this paper, we propose an emotion generation system considering complex emotion based on MaC model using neural networks. In the proposed system, the chaotic neural network and the Kohonen Feature Map (KFM) associative memory are used in the Emotion Generator of the MaC model. The proposed system makes use of the probabilistic association ability of the KFM associative memory in order to generate different emotions for same external input. And, the proposed system makes use of the dynamic association ability of the chaotic neural network in order to generate emotions based on its history. Moreover, the proposed model can deal with not only basic emotions but also complex emotions.

Keywords: Emotion Generation System, Complex Emotion, MaC Model, Chaotic Neural Network, Kohonen Feature Map Associative Memory.

1 Introduction

In various digital mechanical pets and human interfaces, not only autonomous actions but also emotions are introduced based on conceptual model. In most of emotion systems, emotions are generated based on only external input. However, internal emotions are not considered in these systems.

As the model which considers not only external input but also internal emotions, the emotion generation system based on MaC (Mind and Consciousness) model using neural networks[1] has been proposed. This model is based on the MaC model[2] which is the conceptual model of mind and consciousness. The MaC model has (1) mind mechanism which uses emotions for value judgment and (2) consciousness mechanism which processes selective attention and reflection. However, the emotion generation system based on MaC model using neural networks[1] can deal with only basic emotions.

In this paper, we propose the emotion generation system considering complex emotion based on MaC model using neural networks. In the proposed system, the chaotic neural network[3] and the Kohonen Feature Map (KFM) associative

© Springer-Verlag Berlin Heidelberg 2013

memory[4] are used in the Emotion Generator of the MaC model. The proposed system makes use of the probabilistic association ability of the KFM associative memory (which is based on the self-organizing map[5]) in order to generate different emotions for same external input. And, the proposed system makes use of the dynamic association ability of the chaotic neural network in order to generate emotions based on its history. Moreover, the proposed model can deal with not only basic emotions but also complex emotions.

2 Emotion Generation System Considering Complex Emotion Based on MaC Model with Neural Networks

2.1 Outline

In the proposed system, the chaotic neural network[3] and the Kohonen Feature Map (KFM) associative memory[4] are used in the Emotion Generator of the MaC model[2]. The proposed system makes use of the probabilistic association ability of the KFM associative memory in order to generate different emotions for same external input. And, the proposed system makes use of the dynamic association ability of the chaotic neural network in order to generate emotions based on its history. Moreover, the proposed model can deal with not only basic emotions but also complex emotions.

Figure 1 shows the architecture of the proposed system.

2.2 Emotion Model

In the proposed system, the basic emotion model proposed by Plutchik[6] is used as the emotion model. In this model, "anger", "anticipation", "joy", "trust", "fear", "surprise", "sadness" and "disgust" are the basic emotions (See Fig.2). In the proposed system, not only these basic emotions, but also complex emotions such as "love", "optimism", "aggressiveness", "contempt", "remorse", "disapproval", "awe" and "submission" are also considered.

2.3 Emotion in Proposed System

In the proposed system, the emotion value for eight basic emotions at the time t ($E_e(t)$ $(0 \leq E_e(t))$) and the emotion value for eight complex emotions at the time t ($E_c(t)$ $(0 \leq E_c(t))$) are defined. Here, e and c are given by

$$e = \{joy,\ anticipation,\ trust,\ anger,\ fear,\ sadness,\ disgust,\ surprise\} \quad (1)$$

$$c = \{love,\ optimism,\ aggressiveness,\ contempt,\ remorse,\ disapproval,\ awe,\ submission\}. \quad (2)$$

In the proposed system, the positive and negative moods $M_p(t)$ and $M_n(t)$ are defined based on the eight basic emotions, and are given by

$$M_p(t) = (E_{joy}(t-1) + E_{anticipation}(t-1) + E_{trust}(t-1))/3 \quad (3)$$

$$M_n(t) = (E_{anger}(t-1) + E_{fear}(t-1) + E_{sadness}(t-1) + E_{disgust}(t-1))/4. \quad (4)$$

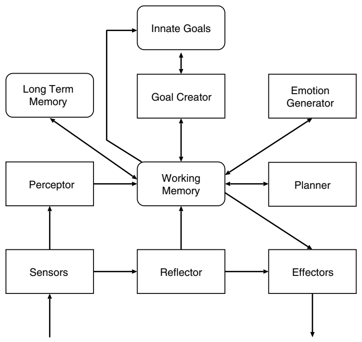


Fig. 1. Architecture of Proposed System

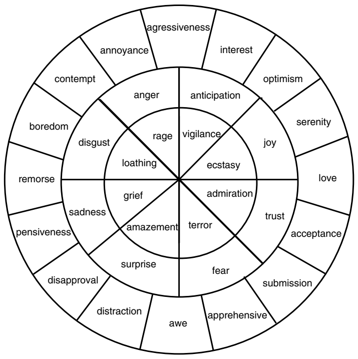


Fig. 2. Plutchik's Basic Emotion Model

The emotion value for surprise $E_{suprise}$ is not used in the calculation of moods because surprise is not categorized into neither positive or negative. If the positive mood is large, the positive emotions (joy, anticipation, trust) become easy to be generated. In contrast, if the negative mood is large, the negative emotions (anger, fear, sadness, disgust) become easy to be generated.

2.4 Function of Each Module

(1) Sensors

The Sensors detect stimuli such as actions and words from the environment to extract physical features. This module sends the information on the received sensory input ($\boldsymbol{I}(t)$) to the Reflector and the Receptor.

(2) Reflector

In the Reflector, whether the input has an urgency is judged. If the input has an urgency, a reflex action (avoidance behavior) is generated. If the urgency level $D(t)$ is larger than the threshold, an avoidance behavior is generated as the reflex action, and the information is sent to the Effectors.

(3) Receptor

In the Receptor, the input stimulus is recognized based on the input information $\boldsymbol{I}(t)$.

In the proposed system, the external input is recognized as pain/pleasantness/like/hate. The input information $\boldsymbol{I}(t)$ and the recognized strength of the stimulus $S_s(t)$ are sent to the Working Memory. Here, s is given by

$$s = \{pain,\ pleasantness,\ like,\ hate\}. \tag{5}$$

(4) Innate Goals

In the proposed system, a desire to want to fill hungry stomach/recover from fatigue/avoid from emergency is set as the innate goal. In the Innate Goals, the desire level for the innate goal $G_{in}(t)$ is calculated based on the previous actions $\boldsymbol{O}(t-1)$ and the urgency level $D(t)$. Here, in is the innate goal and is given by

$$in = \{hungry,\ fatigue,\ urgency\}. \tag{6}$$

(5) Goal Creator

In the Goal Creator, (a) empirical goal for external stimulus, (b) empirical goal for emotion and (c) empirical goal for mood are calculated. From these empirical goal and the desire level for the innate goal, the current goal is set.

(a) Empirical Goal for External Stimulus

As the empirical goal for the external stimulus, a desire to want to avoid pain stimulus/get pleasantness/get favorite things/avoid hate is considered. The desire level for the empirical goal g_s on the external stimulus s at the time t, $G_s(t)$ is calculated based on the desire level for the stimulus s based on the stimulus strength ($G_s^S(t)$) and the desire level for the stimulus s based on the long term memory ($G_s^{LM}(t)$).

(b) Empirical Goal for Emotion

As the empirical goal for the emotion, a desire to want to each emotion is considered. The desire level for the empirical goal g_e on the emotion e at the time t, $G_e(t)$ is calculated based on the desire level for the emotion e ($G_e^E(t)$) and the desire level based on the long term memory ($G_e^{LM}(t)$).

(c) Empirical Goal for Mood

As the empirical goal for the mood, a desire to want to positive or negative emotion is considered. The desire level for the empirical goal g_M on the mood at the time t, $G_M(t)$ is calculated based on the desire level for the mood ($G_M^M(t)$) and the desire level based on the long term memory ($G_M^{LM}(t)$).

(d) Goal Setting

The goal at the time t, $g(t)$ is set based on the desire level for innate goals and empirical goals.

(6) Emotion Generator

In the Emotion Generator, the emotion is generated by the chaotic neural network[3] and the KFM associative memory[4], and the emotion value and the mood are calculated. The proposed system makes use of the probabilistic association ability of the KFM associative memory in order to generate different emotions for same external input. And, the proposed system makes use of the dynamic association ability of the chaotic neural network in order to generate emotions based on its history.

(a) Calculation of Mood

Here, the mood is calculated based on the emotion value at the previous time. The positive mood at the time t ($M_p(t)$) and the negative mood at the time t ($M_n(t)$) are calculated by Eqs.(3) and (4). The mood at the time t ($M(t)$) is given by

$$M(t) = \begin{cases} M_p(t), & (|M_p(t) - M_n(t)| > \theta_M \text{ and } M_p(t) > M_n(t)) \\ 0, & (|M_p(t) - M_n(t)| \leq \theta_M) \\ -M_n(t), & (|M_p(t) - M_n(t)| > \theta_M \text{ and } M_p(t) < M_n(t)) \end{cases} \tag{7}$$

where θ_M is the threshold for the mood.

(b) Calculation of Emotion Value

The emotion value for the emotion e at the time t, $E_e(t)$ is given by

$$E_e(t) = f^E\Bigg(E_e(t-1) + M_e(t) + N_e(t) - F_e + W_e(t)\bigg(T_e(t)\Big(\sum_{in\in N} G_{in}(t) + \sum_{\{s,e,M\}\in N} G_{\{s,e,M\}}(t)\Big)\bigg)\Bigg) \quad (8)$$

$$M_e(t) = \begin{cases} T^1_{M_E}, & (m(t) = p \text{ and } e(t) = e_p(t) \text{ or } m(t) = n \text{ and } e(t) = e_n(t)) \\ T^2_{M_E}, & (m(t) = p \text{ and } e(t) = e_n(t) \text{ or } m(t) = n \text{ and } e(t) = e_p(t)) \\ 0.0, & (\text{otherwise}) \end{cases} \quad (9)$$

where p is the positive mood, n is the negative mood, $e_p(t)$ is the positive emotion, $e_n(t)$ is the negative emotion, and $T^1_{M_E}$ $(0 < T^1_{M_E} < 0.1)$ and $T^2_{M_E}$ $(-0.1 < T^2_{M_E} < 0)$ are the coefficients of mood for emotion generation. $N_e(t)$ is the coefficient for the emotion which is generated in the neural network at the time t and is given by

$$N_e(t) = K_e(t) + C_e(t) \quad (10)$$

where $C_e(t)$ $(0 < T_{C_e}(t) < 0.5)$ is the coefficient which is determined based on the emotion generated in the chaotic neural network for the emotion generation. And, $K_e(t)$ is the coefficient which is determined based on the emotion generated in the KFM associative memory for the emotion generation and is given by

$$K_e(t) = \sum_e T^e_K \quad (11)$$

$$T^e_K = \begin{cases} E^e_k, & (e = K_o) \\ -E^e_K/2, & (e = K_{o'}) \\ 0, & (\text{otherwise}) \end{cases} \quad (12)$$

where K_o is the recalled emotion by the KFM associative memory, E^e_K is its emotion value, and $K_{o'}$ is the emotion in the opposite position in the Plutchik model.

And F_e is the inhibitory coefficient for the emotion e and is given by

$$F_e = f^{E_c}_e\left(E_e(t-1)\right) \quad (13)$$

$$f^{E_c}_e(u) = \begin{cases} F^4_e, & (\theta^3_e \le u) \\ F^3_e, & (\theta^2_e \le u < \theta^3_e) \\ F^2_e, & (\theta^1_e \le u < \theta^2_e) \\ F^1_e, & (u < \theta^1_e) \end{cases} \quad (14)$$

where F^1_e, F^2_e, F^3_e and F^4_e $(F^4_e > F^3_e > F^2_e > F^1_e)$ are the coefficients.

$W_e(t)$ is the coefficient from the recalled emotion e by the neural network. $T_e(t)$ is the coefficient for the emotion e and is given by

$$T_e(t) = f^{E_T}_e\left(\sum_{n\in N} G_n(t) - G_n(t-1), E_e(t-1)\right) \quad (15)$$

$$f_e^{E_T}(u_1, u_2) = \begin{cases} \mathrm{sgn}(u_1)T_e^4, & (\theta_e^3 \leq u_2) \\ \mathrm{sgn}(u_1)T_e^3, & (\theta_e^2 \leq u_2 < \theta_e^3) \\ \mathrm{sgn}(u_1)T_e^2, & (\theta_e^1 \leq u_2 < \theta_e^2) \\ \mathrm{sgn}(u_1)T_e^1, & (u_2 < \theta_e^1) \end{cases} \tag{16}$$

where θ_e^1, θ_e^2 and θ_e^3 are the thresholds, and T_e^1, T_e^2, T_e^3 and T_e^4 is update values for the emotion e.

In Eq.(8), $f^E(\cdot)$ is given by

$$f^E(u) = \begin{cases} u, & (u > 0) \\ 0, & (u \leq 0). \end{cases} \tag{17}$$

(c) Output of Complex Emotions

The emotions is output based on the emotions which are recalled in the neural networks and their emotion value calculated in (b).

Step 1 : If the output of the KFM associative memory corresponds to the complex value c, c is set to the candidate of output emotion.

Step 2 : If the difference between emotion values of two adjacent basic emotions in the Plutchik model[6] is smaller than the threshold, the complex emotion c between these two basic emotions is is set to the candidate of output emotion.

Step 3 : The complex emotion value E_c is calculated based on the emotion value for the complex emotion selected in **Step 1** and **Step 2**.

Step 4 : If the emotion value is larger than the threshold, the complex emotion is generated.

(7) Planner

In the Planner, the action is selected based on the emotions as similar as in the conventional system[1].

(8) Effector

In the Effector, the action which is selected in the Reflector or the Planner is carried out.

(9) Long Term Memory Part

In the Long Term Memory, the information from the other modules are memorized. Here, the input information, the emotions and their emotion values, the complex emotions and their emotion values, the mood, the goal, the desire level and the action are memorized.

(10) Working Memory

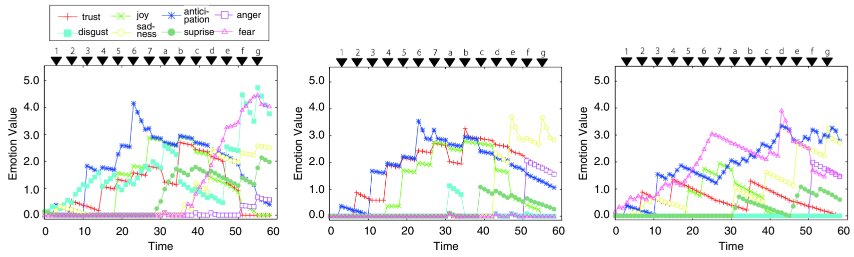
In the Working Memory, the information from the other modules are memorized temporarily.

3 Experiment Results

3.1 Emotion Generation

Here, the emotions were generated by the proposed system, the method without internal emotions and the method without mood.

As the stimuli which makes to generate the positive emotions,



(a) Proposed System. (b) Without Internal Emotions (c) Without Mood

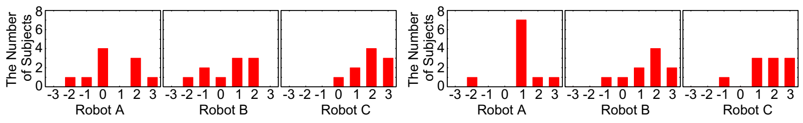
Fig. 3. Variation of Emotion Values

1. “Hello.” ⇒ 2. “Let’s play together.” ⇒ 3. show favorite thing ⇒ 4. “Here you are.” ⇒ 5. “You are happy, aren’t you?” ⇒ 5. “Do you want candies?” ⇒ 6. pat ⇒ 7. “Good.”

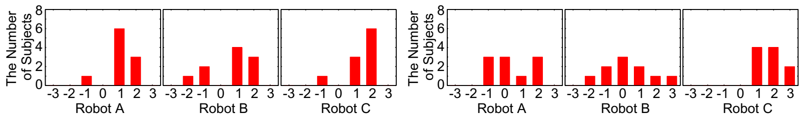
were given. And as the stimuli which males to generate the negative emotions,

a. back ⇒ b. “Bye bye.” ⇒ c. “Don’t come here.” ⇒ d. hit strong ⇒ e. “Get away.” ⇒ f. show least favorite thing ⇒ g. “Are you sad?”

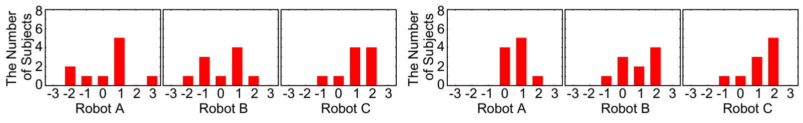
were given.



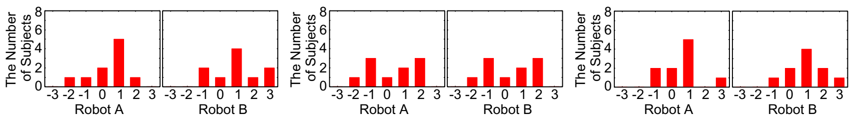
[Q1] hostile ↔ friendly [Q2] action looks to have no aim ↔ action looks to have aim



[Q3] unnatural ↔ natural [Q4] hard to communicate ↔ easy to communicate



[Q5] looks like machine ↔ looks like creature [Q6] looks like non-human ↔ looks like human



[Q7] unnatural ↔ natural [Q8] looks like machine ↔ looks like creature [Q9] looks like non-human ↔ looks like human

Fig. 4. Questionnaire Results

Figure 3(a) shows the variation of emotion values in the proposed system and (b) and (c) show the variation of emotion values in the method without internal emotions and mood. In these figures, the numbers or characters at the top show the action and words. From these results, we can see that the proposed system can generate emotions naturally independent only external input.

3.2 Experiment Using Robot

Here, we examined using the robot based on the proposed system (Robot A), the robot based on the system without internal emotions (Robot B) and the robot operated manually (Robot C). We used the robot build of the LEGO mindstorms NXT. In this experiment, subjects give some actions through sensors or words through GUI and the robot actions to the input. We used the SD (semantic differential) method and the subjects evaluated the behavior of the robot in 7 levels ($-3 \sim 3$). (Q1∼6 for Robot's actions, Q7∼ 9 for Generated Emotions)

[Q1]	hostile	$\leftrightarrow$	friendly
[Q2]	action looks to have no aim	$\leftrightarrow$	action looks to have aim
[Q3][Q7]	unnatural	$\leftrightarrow$	natural
[Q4]	hard to communicate	$\leftrightarrow$	easy to communicate
[Q5][Q8]	looks like machine	$\leftrightarrow$	looks like creature
[Q6][Q9]	looks like non-human	$\leftrightarrow$	looks like human

4 Conclusions

In this paper, we have proposed the emotion generation system considering complex emotion based on MaC model using neural networks. We carried out computer experiments, and confirmed that the proposed system can realize autonomous and human-like emotion generation. Moreover, we confirmed that the robot based on the proposed system can decide actions as similar as in the robot operated manually.

References

1. Hirozawa, K., Osana, Y.: Emotion generation system based on MaC model with neural networks. In: Proceedings of IEEE International Conference on System, Man and Cybernetics, San Antonio (2009)
2. Ushida, H., Hirayama, Y., Nakajima, H.: Emotion model for life-like agent and its evaluation. In: Proc. AAAI 2008: Fifth National Conference on Artificail Intelligence, Madison, pp. 62–69 (1998)
3. Aihara, K., Takabe, T., Toyoda, M.: Chaotic neural networks. Physics Letter A 144(6-7), 333–340 (1990)
4. Sato, H., Osana, Y.: Variable-sized Kohonen feature map probabilistic associative memory. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 371–378. Springer, Heidelberg (2012)
5. Kohonen, T.: Self-Organizing Maps. Springer (1994)
6. Plutchik, R.: A general psychoevolutionary theory of emotion. In: Plutchik, R., Kellerman, H. (eds.) Emotion: Theory, Research, and Experience, pp. 3–33. Academic Press (1980)

Neuro-Optimal Controller for Robot Manipulators

Mohammed Boukens and Abdelkrim Boukabou

Department of Electronics, Jijel University, 98 Ouled Aissa, Jijel 18000, Algeria

Abstract. This paper presents an algorithm for continuous-time quadratic optimization with neural compensation of motion control. A simpler reformulation explicit solution to the Hamilton-Jacobi-Bellman equation for optimal control of rigid robot motion is found by solving an algebraic Riccati matrix equation. The system stability is investigated according to Lyapunov function theory and it is shown that global asymptotic stability holds in the case of known system model. It is also shown how optimal control and neural control may act in concert in the case of unknown system model. The neural algorithm is derived from Lyapunov stability analysis, so that both system-tracking stability and error convergence can be guaranteed in the closed-loop system. Experimental and simulation results from a two-link robot manipulator show the satisfactory performance of the proposed control scheme.

Keywords: Optimal control, neural networks, robotic control.

1 Introduction

Control of linear systems is now a well-proficient field. The tools of linear algebra allow obtaining for these systems, control laws possessing properties of stability and optimality. Over the last decade, a lot of research effort has been put into the design of sophisticated control strategies for robots. A number of model-based robot control methods have been proposed such as computed torque control and PD with gravity compensation [1]. These methods require exact knowledge of the nonlinear robot dynamics, which, in practice is generally not available. As the manipulator is a multivariable non-linear coupling dynamic system with uncertainties, it is difficult to obtain an accurate mathematical model so that classical or modern control laws can be applied. Thus, conventional controllers cannot effectively control the motion of a robot due to complications of these nonlinear effects.

To solve this problem, during the last few years, it has emerged as a major research area with ramifications in artificial intelligence. Fuzzy part with a set of tunable parameters is employed to approximate lumped uncertainty due to parameters variations, unmodeled dynamics and so on in robotic manipulators [2]. Another solution is to use the universal approximation of nonlinear functions by artificial neural networks, such as the approach proposed by Lewis [3], and also discussed in [4], which allow approaching any nonlinear function. It is this property that motivates their use for the production of nonlinear control systems.

© Springer-Verlag Berlin Heidelberg 2013

Perez *et al.* [5] proposed a trajectory tracking error using PID control law for two-link robot manipulator. The neural network controller is trained by a training rule developed to minimize a cost function.

In this paper, an optimal control strategy with neural compensation is proposed in which plant dynamics is learned by a neural network. The online learned neural network is used to adaptively estimate nonlinear uncertainties and improve performance in the face of unknown nonlinearities by adding nonlinear effects to the optimal controller, yielding a controller that can tolerate a wider range of uncertainties. The proposed optimal controller adapts itself so that the learning rate and the time parameter track the dynamical behavior. The real time results showed the effectiveness of the proposed method with respect to desired trajectory tracking and disturbance rejection.

The paper is organized as follows. In Section 2, robot dynamics and problem formulation is given. In Section 3, the proposed neuro-optimal controller is detailed. In Section 4, an experimental example is realized that illustrates a concrete implementation for the problem of trajectory tracking of a two-link manipulator. Finally, conclusion is drawn in Section 5.

2 Robot Arm Dynamics and Problem Statement

The dynamics in joint space of a serial-chain n-link robot manipulator can be written as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + \tau_d = \tau, \tag{1}$$

where the position coordinates $q(t) \in \mathbb{R}^n$ with associated velocities $\dot{q}(t)$ and accelerations $\ddot{q}(t)$ are controlled with the external driving forces $\tau(t) \in \mathbb{R}^n$. The moment of inertia $M(q) \in \mathbb{R}^{n\times n}$, the Coriolis/centripetal $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ and gravitational forces $G(q) \in \mathbb{R}^n$ all vary along the trajectories. $\tau_d \in \mathbb{R}^n$ represents the external and unknown disturbances.

We assume that the positions $q(t)$ and velocities $\dot{q}(t)$ are available for measurement. Moreover, we assume that the external control torque $\tau(t)$ is available as the control input and that the matrices $M(q)$ and $C(q,\dot{q})$ are of structure and satisfy the following properties:

Property 1: The inertia matrix $M(q)$ is symmetric positive, *i.e.,* $M(q) = M^T(q) > 0$.

Property 2: The matrix $N(q,\dot{q}) = \dot{M}(q) - 2C(q,\dot{q})$ is skew-symmetric, *i.e.,* $x^T N(q,\dot{q})x = 0; \forall x \in \mathbb{R}^n$.

Given a desired trajectory $q_d(t) \in \mathbb{R}^n$, the tracking errors are defined as $e(t) = q_d(t) - q(t)$ and $\dot{e}(t) = \dot{q}_d(t) - \dot{q}(t)$, or in the state space representation as $x(t) = \left(\dot{e}^T(t)\ e^T(t)\right)^T$.

We introduce the following state space transformation of $x(t)$:

$$z(t) = Tx(t) = \begin{pmatrix} T_1 & T_2 \\ 0_{n\times n} & I_{n\times n} \end{pmatrix} \begin{pmatrix} \dot{e}(t) \\ e(t) \end{pmatrix}, \tag{2}$$

where $T > 0$, $T_1, T_2 \in \mathbb{R}^{n\times n}$ and $z(t) \in \mathbb{R}^{2n}$.

Now define a control-input torque as

$$\tau(t) = M(q)(\ddot{q}_d + T_1^{-1}T_2\dot{e}) + C(q,\dot{q})(\dot{q}_d + T_1^{-1}T_2 e) + G(q) - T_1^{-1}Ru(t), \quad (3)$$

with $R = R^T \in \mathbb{R}^{n\times n}$ positive definite and $u(t) \in \mathbb{R}^n$ an auxiliary control input to be optimized. The closed-loop system (1) becomes

$$M(q)(T_1\ddot{e}(t) + T_2\dot{e}(t)) = -C(q,\dot{q})(T_1\dot{e}(t) + T_2 e(t)) + Ru(t). \quad (4)$$

And the following augmented system is obtained:

$$\begin{aligned}\dot{z}(t) &= \begin{pmatrix} -M^{-1}(q)C(q,\dot{q}) & 0_{n\times n} \\ T_1^{-1} & -T_1^{-1}T_2 \end{pmatrix} z(t) + \begin{pmatrix} I_{n\times n} \\ 0_{n\times n} \end{pmatrix} M^{-1}(q)Ru(t), \\ &= A(q,\dot{q})z(t) + B(q)u(t), \end{aligned} \quad (5)$$

with $A(q,\dot{q}) \in \mathbb{R}^{2n\times 2n}$, $B(q) = \hat{B}M^{-1}(q)R \in \mathbb{R}^{2n\times 2n}$ and $\hat{B} = \begin{pmatrix} I_{n\times n} \\ 0_{n\times n} \end{pmatrix}$.

The control problem is formulated as a quadratic optimization problem $J(z,u) = \int_{t_0}^{\infty} L(z,u)dt$ with the Lagrangian $L(z,u) = \frac{1}{2}(z^TQz + u^TRu)$.

A necessary and sufficient condition for $\bar{u}\ (t)$ to minimize (4) subject to (5) is that there exists a function $V(z,t)$ satisfying the following Hamilton-Jacobi-Bellman (HJB) equation

$$-\frac{\partial V}{\partial t} = \min_u \left\{ L(z,u) + \left(\frac{\partial V}{\partial z}\right)^T \dot{z} \right\}, \quad (6)$$

where $V(z,t)$ satisfies the partial differential equation

$$\dot{V}(t,z) = \frac{\partial V}{\partial t} + \left(\frac{\partial V}{\partial z}\right)^T \dot{z} = -L(z,\bar{u}) = -\frac{1}{2}(z^TQz + \bar{u}^TR\bar{u}), \quad (7)$$

with Q and R positive definite diagonal matrices and $\hat{B}^T = (I_{n\times n}\ 0_{n\times n})$. The optimal feedback control law $\bar{u}\ (t)$ that minimized $J(z,u)$ is given by

$$\bar{u}(t) = -\hat{B}^T z(t) = -\left(T_1\ T_2\right) x(t) = -\left(T_1\dot{e}(t) + T_2 e(t)\right). \quad (8)$$

3 Neural-Law Design

Given $x = [x_1, x_2, \ldots, x_{N_1}]^T \in \mathbb{R}^{N_1}$, a three-layer neural net has a net output given by

$$y_i = \sum_{j=1}^{N_2} w_{ij} \left[\varphi \left(\sum_{k=1}^{N_1} v_{jk} x_k \right) \right]; \quad i = 1,..,N_3, \quad (9)$$

with $\varphi(\cdot)$ the activation function, v_{jk} the first-to-second layer interconnection weights, w_{ij} the second-to-third layer interconnection weights and N_2 the number of hidden-layer neurons. The proposed neuro-optimal controller scheme is shown in Fig. 1.

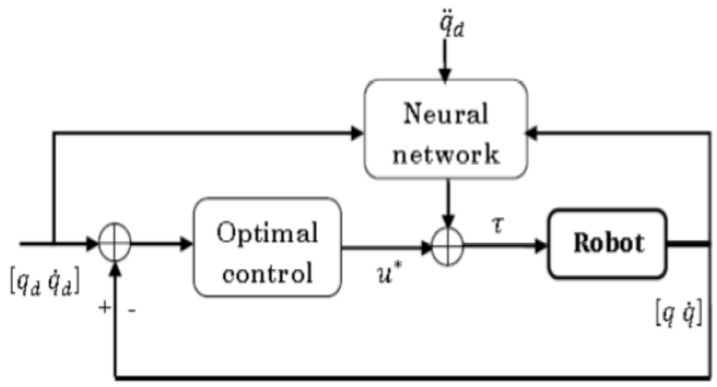


Fig. 1. Neuro-optimal controller scheme

The appropriate external torque to apply to the robotic system is calculated according to the control input (3) as follows

$$\tau(t) = f(x) - T_1^{-1}R\bar{u}(t) = f(x) + R\dot{e}_r(t), \tag{10}$$

where

$$f(x) = M(q)\ddot{q}_r(t) + C(q,\dot{q})\dot{q}_r(t) + G(q), \tag{11}$$

and we may define

$$\begin{cases} x = \left[e^T\ \dot{e}^T\ q_d^T\ \dot{q}_d^T\ \ddot{q}_d^T\right]^T,\ \lambda = T_1^{-1}T_2 = Q_1^{-1}Q_2, \\ \ddot{q}_r = \ddot{q}_d + \lambda\dot{e}\ ,\quad \dot{e}_r = \dot{e} + \lambda e. \end{cases} \tag{12}$$

Then the nonlinear robot function $f(x)$ can be assumed as the following model

$$f(x) = W^T\varphi(V^Tx), \tag{13}$$

where V^T contains both the weights of the first-to-second layer connections, W^T contains both the weights of the second-to-third layer connection and $\varphi(\cdot)$ is the nonlinear activation function chosen as $\varphi(\cdot) = \tanh(\cdot)$.

We will determine later the laws of weight adjustment NN from Lyapunov stability analysis for estimating online the nonlinear function such that the closed loop system is stable. The estimate $\hat{f}(x)$ of $f(x)$ can be written as

$$\hat{f}(x) = \hat{W}^T\varphi(\hat{V}^Tx), \tag{14}$$

with $\hat{W}$ and $\hat{V}$ represent the estimates of the target weight values. Thus, the external torque (10) is now given by

$$\tau(t) = \hat{W}^T\varphi(\hat{V}^Tx) + R\dot{e}_r(t). \tag{15}$$

The closed-loop error dynamics become

$$M(q)\ddot{e}_r + (C(q,\dot{q}) + R)\dot{e}_r = W^T\varphi(V^Tx) - \hat{W}^T\varphi(\hat{V}^Tx) + \varepsilon(x) + \tau_d. \tag{16}$$

It is important to note that the NN reconstruction error $\varepsilon(x) = f(x) - \hat{f}(x)$ and the robot disturbances τ_d have exactly the same influence as disturbances in the error system. Let us define the hidden-layer output error for a given x as $\tilde{\varphi} = \varphi - \hat{\varphi} = \varphi(V^T x) - \varphi(\hat{V}^T x)$ and the weight estimation errors as $\tilde{W} = W - \hat{W}$, $\tilde{V} = V - \hat{V}$, then, the Taylor series expansion for a given x may be written as $\varphi(V^T x) = \varphi(\hat{V}^T x) + \varphi\prime(\hat{V}^T x)\tilde{V}^T x + O(\tilde{V}^T x)^2$ where $\varphi\prime$ is the Jacobean.

Setting $\hat{\varphi}\prime = \varphi\prime(\hat{V}^T x)$, we have

$$\tilde{\varphi} = \hat{\varphi}\prime\tilde{V}^T x + O(\tilde{V}^T x)^2. \tag{17}$$

Substituting (17) into (16) gives

$$M(q)\ddot{e}_r + (C(q,\dot{q}) + R)\dot{e}_r = \tilde{W}^T(\hat{\varphi} - \hat{\varphi}\prime\hat{V}^T x) + \hat{W}^T\hat{\varphi}\prime\tilde{V}^T x + w, \tag{18}$$

where the disturbances in the error system is $w = \tilde{W}^T\hat{\varphi}\prime V^T x + W^T O(\tilde{V}^T x)^2 + \varepsilon(x) + \tau_d$.

Theorem 1. *Consider that the control action is provided by the optimal controller $\bar{u}$ (8), and that the gradient based training for neural network weights is given by*

$$\begin{cases} \dot{\hat{W}} = F(\hat{\varphi} - \hat{\varphi}'\hat{V}^T x)\dot{e}_r^T - kF\|\dot{e}_r\|\hat{W}, \\ \dot{\hat{V}} = Gx(\hat{\varphi}'^T\hat{W}\dot{e}_r)^T - kG\|\dot{e}_r\|\hat{V}, \end{cases} \tag{19}$$

with $k \in \mathbb{R}^+$, $F = F^T > 0$ and $G = G^T > 0$. Then the errors $e(t)$ and $\dot{e}(t)$ are uniformly ultimately bounded (UBB). Moreover, the errors can be made arbitrarily small by adjusting weighting matrices $\hat{W}$ and $\hat{V}$.

Proof. Consider the following Lyapunov function

$$L = \frac{1}{2}\dot{e}_r^T M(q)\dot{e}_r + \frac{1}{2}tr(\tilde{W}^T F^{-1}\tilde{W}) + \frac{1}{2}tr(\tilde{V}^T G^{-1}\tilde{V}). \tag{20}$$

where $tr(\cdot)$ is the trace operator. Computing the time derivative of the function, we have

$$\dot{L} = \dot{e}_r^T M(q)\ddot{e}_r + \frac{1}{2}\dot{e}_r^T\dot{M}(q)\dot{e}_r + tr(\tilde{W}^T F^{-1}\dot{\tilde{W}}) + tr(\tilde{V}^T G^{-1}\dot{\tilde{V}}), \tag{21}$$

By substituting Eq. (18), the learning rule (19) and from property 2, then we obtain

$$\dot{L} = -\dot{e}_r^T R\dot{e}_r + k\|\dot{e}_r\| tr\tilde{Z}^T\left(Z - \tilde{Z}\right) + \dot{e}_r^T w, \tag{22}$$

where $\tilde{Z} = \begin{pmatrix} \tilde{W} & 0 \\ 0 & \tilde{V} \end{pmatrix}$. We know that

$$\begin{cases} tr\tilde{Z}^T\left(Z - \tilde{Z}\right) = tr\left(\tilde{Z}, Z\right) - \|\tilde{Z}\|_F^2 \leq \|\tilde{Z}\|_F\|Z\|_F - \|\tilde{Z}\|_F^2, \\ \lambda_{min}\{R\}\|\dot{e}_r\|^2 \leq \dot{e}_r^T R\dot{e}_r, \end{cases} \tag{23}$$

with $\|A\|_F$ the Frobenius norm and $\lambda_{min}\{R\}$ the minimum eigenvalue. If R is taken as a diagonal matrix, then $\lambda_{min}\{R\}$ is simply the smallest gain element.

In addition, we consider that the target weights and the disturbance term are bounded, *i.e.*, $\|Z\|_F \leq Z_b \in \mathbb{R}^+$ and $\|w\| \leq w_b \in \mathbb{R}^+$, respectively. These considerations will be true in every practical situation because the power into the system is bounded. Hence, the time derivative of the Lyapunov function can be rewritten by

$$\dot{L} \leq -\|\dot{e}_r\| \left(\lambda_{min}\{R\}\|\dot{e}_r\| + k\|\tilde{Z}\|_F\left(\|\tilde{Z}\|_F - Z_b\right) - w_b\right), \tag{24}$$

which is negative as long as the term in braces is positive. Thus

$$\dot{L} \leq -\|\dot{e}_r\| \left[\lambda_{min}\{R\}\|\dot{e}_r\| + k\left(\|\tilde{Z}\|_F - \frac{Z_b}{2}\right)^2 - \frac{k}{4}Z_b^2 - w_b\right] \tag{25}$$

That is guaranteed negative as long as one of the two following inequalities holds

$$\|\dot{e}_r\| > \frac{1}{\lambda_{min}\{R\}}\left(\frac{k}{4}Z_b^2 + w_b\right) = b_e \text{ or } \|\tilde{Z}\|_F > \frac{Z_b}{2} + \sqrt{\frac{Z_b^2}{4} + \frac{w_b}{k}} = b_z, \tag{26}$$

where b_e and b_z are the convergence regions associated to $\dot{e}_r$ and $\tilde{Z}$, respectively. Thus, $\dot{L}$ is negative outside a compact set. Note that b_e can be kept small by adjusting the design parameter $\lambda_{min}\{R\}$ which ensures that $\dot{e}_r(t)$ stays in the compact set. We can choose $\lambda_{min}\{R\} \gg 1$.

According to the standard Lyapunov theorem extension, this demonstrates the uniformly ultimate boundedness (UUB) of both $\|\dot{e}_r\|$ and $\|\tilde{Z}\|_F$. In both cases, there is a convergence of tracking errors. This completes the proof. ■

4 Experimental Case Study

The robot manipulator of Fig. 2 is an experimental 2-DOF lightweight mechanical construction moving in the horizontal plane. It consists of two aluminum links, both actuated by a current-driven DC motor with gearbox. The robot manipulator is connected to a control board based dsPIC ($117MHz$ from Microchip), which constitutes the I/O interface to the robot. The experimental results of the position tracking, the angular velocities, the errors $e_1(t)$, $e_2(t)$, and the control signal $\tau_1(t)$, $\tau_2(t)$ are shown in Figs. 3 (a–h), respectively.

The control input is computed from the optimal-neural control laws (15) and (19) with four neurons in hidden-layer and sampling time $T = 1ms$. From Fig.

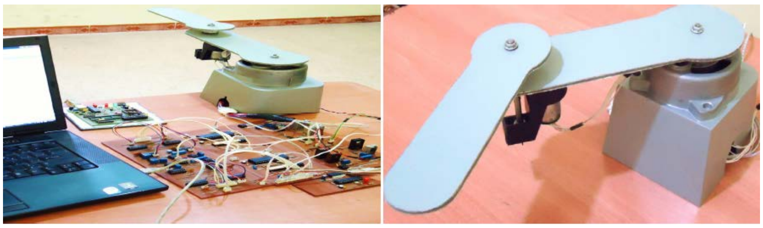
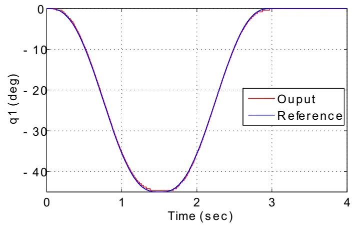
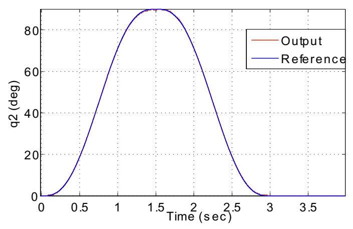


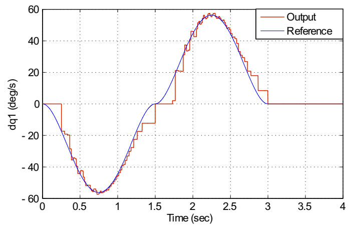
Fig. 2. Experimental robot manipulator



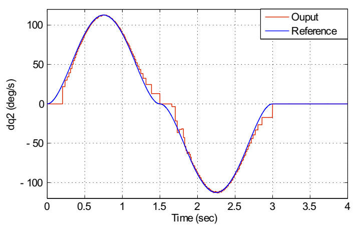
(a) Tracking position for joint 1.



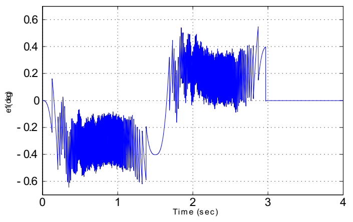
(b) Tracking position for joint 2.



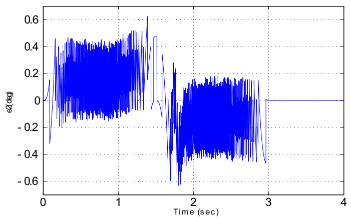
(c) Tracking velocity for joint 1.



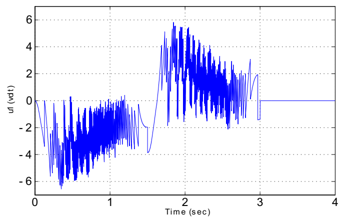
(d) Tracking velocity for joint 2.



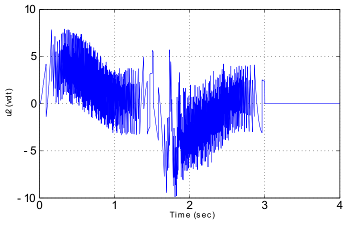
(e) Tracking error for joint 1.



(f) Tracking error for joint 2.



(g) Control signal for joint 1.



(h) Control signal for joint 2.

Fig. 3. The tracking position, velocities and controls signals

3 (e, f), it can be seen that the absolute maximum position tracking error for joint 1 is about 0.6 deg (0.01 rad) and it is about 0.6 deg (0.01 rad) for joint 2. Then, we conclude that the proposed control law gives outstanding performance in terms of error with respect to the resolution of the sensors used (respectively 0.4 and 0.48 degree) and thus the algorithm is capable of compensating unknown dynamics in the robotic manipulator with quite good results.

Remark 2. *The weights $\hat{W}$ and $\hat{V}$ may be initialized as zero takes the NN out of the circuit and leaves only the outer tracking loop in Fig. 1. It is well known that the optimal controller term in (20) can then stabilize the robot arm on an interim basis until the NN begins to learn. This means that there is no off-line learning phase for this NN controller. For practical purposes it is only necessary to select $\lambda_{min}\{R\}$ large until obtaining a reasonable performance but not yet acceptable in the tracking errors. Then, we can use the learning law for reduce the tracking errors with a minimum of overshoot.*

5 Conclusion

In this paper, an online neural network based optimal controller for a robot manipulator is designed. The main reason of using the proposed control scheme is that many systems in practice, such that robot manipulators are not only nonlinear, but also contain uncertainties due to tolerance variation in the manipulator-link properties, unknown loads, and so on. The proposed neuro-optimal controller avoided the need system's model. The experimental results show that this control law gives excellent tracking performance in terms of tracking error with respect to the resolution of the sensors used.

References

[1] Kelly, R., Santibnez, V., Loria, A.: Control of robot manipulators in joint space. Springer, London (2005)
[2] Song, Z., Xinchun, L.: A computed torque controller for uncertain robotic manipulator systems: Fuzzy approach. Fuzzy Sets and Systems 154, 208–226 (2005)
[3] Lewis, F.L.: Neural network control of robot manipulators. Intelligent Systems and their Applications IEEE Expert 11, 64–75 (1996)
[4] Ren, X., Lewis, F.L., Zhang, J.: Neural network compensation control for mechanical systems with disturbances. Automatica 45, 1221–1226 (2009)
[5] Perez, J.P., Perez, J.P., Soto, R., Flores, A., Rodriguez, F., Meza, J.L.: Trajectory tracking error using PID control law for two-link robot manipulator via adaptive neural networks. Procedia Technology 3, 139–146 (2012)
[6] Anderson, B.D.O., Moore, J.B.: Optimal Control: Linear Quadratic Methods. Prentice Hall, New Jersey (1990)

Learning to Walk Using a Recurrent Neural Network with Time Delay

Boudour Ammar[1], Naima Chouikhi[1], Adel M. Alimi[1], Farouk Chérif[2], Nasser Rezzoug[3], and Philippe Gorce[3]

[1] REGIM: REsearch Groups on Intelligent Machines, University of Sfax, National Engineering School of Sfax (ENIS), BP 1173, Sfax, 3038, Tunisia
{boudour.ammar,adel.alimi}@ieee.org

[2] ISSATS, Laboratory of Math Physics, Specials Functions and Applications, LR11ES35, Ecole Supérieure des Sciences et de Technologie, 4002- Sousse- Tunisia
faroukcheriff@yahoo.fr

[3] HandiBio Laboratory, EA 3162, Avenue de l'université, University of South Toulon-Var, BP 20132, 83957 La Garde cedex, France

Abstract. Walking based on gaits is one of the most approved methodologies for walking robot. In this paper, we develop learning strategy for walking biped robot or human based on a self made database using biomechanical capture. This system is provided by a Recurrent Neural Network (RNN) with an internal discrete time delay. The role of the proposed network is the training of human walking data by giving an estimation of the biped's next position at each time and achieve a human-like natural walking. Different architectures of RNN are proposed and tested. In Particular, a comparative study is given and the results of the RNN mixed with extended Kalman filter are illustrated.

Keywords: Human walking, 3D human simulator, Recurrent Neural Network, Biomechanics, e-kalman filter.

1 Introduction

In a few years, robotics has become an important science that is constantly evolving. Within this issue, bipedal locomotion proposals [12] focus on different aspects of robot control. In fact, robots have demonstrated an ability to try to mimic humans in achieving activities in many fields such as industry, home automation, exploration, games and so forth. In all these fields, robots need to displace and move in order to realize their missions. The bipedal locomotion is supposed to be one of the best locomotion systems according to small support surface [2].

Artificial Neural Networks (ANNs) are computational tools, based mainly on the properties of biological neural systems as its network theory revolves around the idea that certain properties of biological neurons can be extracted and applied to simulations [14].

ANNs can be defined as an attempt to somehow imitate the neural network of a human being in a simple way. Although the ANNs can learn in the presence of noisy inputs, they cannot perform to react convincingly and dynamically to a real world situation. If we need to keep in memory the previous states of the network, it would be

© Springer-Verlag Berlin Heidelberg 2013

better to choose the RNN architecture to perform complex tasks ([1] and [9]). Several types of RNN have been used in the past years, for example: cellular neural networks, Hopfield neural networks, Cohen-Grossberg neural networks, bidirectional associative memory neural net-works. Besides, it is well known that there are time delays in the information processing of neurons. So, the RNN with time delays has received much more attention both in theory and in practice ([1], [18], [4]).

In addition, the use of the RNNs can be justified by their dynamism and powerful algorithms of training.

Throughout this paper, we are, mainly, interested in the learning of the human gait to a 3D biped simulator via a RNN. The main contributions of this paper are summarized below:

- Preparation of human walking database based on biomechanic study
- Novel model and architecture of RNN with discrete time delay
- Comparison between some RNN architectures and RNN mixed with e-kalman

By the end of the training, the simulator is able to estimate in each time step the next position during the gait trajectory. A simple comparison proves the effectiveness of a RNN with time delay .

The plan of this paper is organized as follows: in section II, we give an overview about the prepared database of gaits following biomechanic steps. We, also, introduce the RNN model which can be used in the training algorithm and the walking learning system. In Section III, the different architectures of RNN and results are given after training and a comparative study is presented. Finally, section IV draws the conclusions and provides suggestions for future work.

2 Training with RNN

The recurrent neural network is a class of neural network where connections between the units form a cycle. This can lead to an internal dynamic temporal behavior ([10] and [8]). As a Neural Network, the RNN has input, hidden and output layers where each layer has a number of neurons. Besides, in our system, the learning process is supervised and requires a training data set.

2.1 Training Data Set

The learning process consists of two main parts which are training and testing.

In fact, the data set is composed of five walking tests for six subjects. The test capture saves the changes of the joint angles of the different articulations of the two legs.

To do so, a set of distinguishable markers (59 markers) are placed on six human body landmarks. The scene is captured by an opto-electric system composed of a number of calibrated and synchronized cameras [16].

The system of measure consists of some steps according to biomechanical study. In order to establish a relation between the markers, a Human Body Model (HBM) is defined and our own database of gaits is built. Markers are detected on all the camera views and delivered as the input of a particle scheme where every particle encodes an instance of the estimated position of the HBM [17].

So, we save the human movements while walking and we extract the marker's trajectories. The joint trajectories are deducted from the markers position in each instance using some implemented functions that follow a biomechanical study. We mention here that not all the markers are used and only anatomical and technical markers in the lower trunk are implemented because we are concerned up to now with the learning of biped walking. To train the network, the database consists of 10 joint angles at instant *t* and *t+1* as inputs and outputs respectively. Once the database is finished, we can move to the training and test phases.

2.2 The Proposed RNN Model

Many phenomena exhibit a great regularity without being periodic. This is modeled using the notion of pseudo almost periodic functions which allows complex repetitive phenomena to be represented as an almost periodic process plus an ergodic component. Besides, it is well-known that there are time delays in the information processing of the neurons due to various reasons. For instance, the time delays can be caused by the finite switching speed of amplifier circuits in neural networks or deliberately introduced to achieve tasks of dealing with motion-related problems.

The proposed NN model is studied with the aim of achieving human-like performance, especially in the task of human walking. It is an application of the proposed RNNs with time-varying coefficients and discrete delay detailed in [1]. Using the technique of the Euler discretization of the simplified continuous formulation, the approximation of the equation is:

$$S_i(t+1) = S_i(t) + \sum_{j=0}^{n} (c_{i,j}(t) f_j(S_j(t)) + d_{i,j}(t) g_j(S_j(t-\tau))) + J_i. \qquad (1)$$

Where: n is the number of neurons in each network, $S_i(t)$ denotes the state of the ith neuron at time t, f_j and g_j are the activation functions of the jth neuron at time t, $c_{(i,j)}(t)$ and $d_{(i,j)}(t)$ are the connection weights of the jth neuron on the ith neuron, τ is constant time delay and $J_i\,(t)$ is the external bias on the *i*th neuron.

The proposed architecture of the RNN is composed of ten inputs, ten outputs and a variable number of neurons in a hidden layer. We choose the joint angles at instant *t*, as inputs and the joint angles at instant *t+1*, as outputs.

This network aims to obtain a human-like robot behavior and estimates the next values of the different joints.

2.3 BPTT Training Algorithm

In this paper, we adopted the Back Propagation Through Time (BPTT) as a training algorithm. In fact, the RNNs are unfolded in time; which means that they can be transformed from the feedback structure to the feed forward structure [3].

Then, an error is calculated at each unfolding step beginning with the last layer and retro-propagated to the previous layer which propagates its error forth until reaching the first layer.

A database with ten inputs and ten outputs is introduced to the network. The angles of the articulations of the lower trunk at instant *(t)* and *(t+1)* are respectively the inputs and outputs. Then, the data set is divided into trained data set (2400 steps of 8 tests of human subjects) and test data set (300 steps of 1 test of human subject). The last database is used to calculate the performance of the RNN and the error is computed according to equation 2. For predictive analytics, the error function is the sum-of-squared errors [8]. This function is calculated by looking at the squared difference between what the network predicts for each training pattern and the target value, or observed value, for that pattern. The global error is, then, deduced from the equation 3.

$$E_k = \frac{1}{2}(S_d(t+k) - S(t+k))^2. \tag{2}$$

Where: k is the index of the step or sample, $S_d(t+k)$ and $S(t+k)$ are respectively the desired output vector and the output vector of the network in this step.

$$E = \frac{1}{N}\sum E_k. \tag{3}$$

Where: N is the number of samples.

Some parameters, which can be adjusted in the RNN, are increasing the iteration number or/and adding new hidden layers or/and altering the number of neurons in each hidden layer. The feedback between neurons from the input, hidden and output layers is tested and explained below. The update of weights can be performed with e-kalman filter. The last is an algorithm that uses a series of measurements observed over time, containing noise (random variations) and other inaccuracies, and produces estimates of weight that tend to be more precise than those based on BPTT algorithm alone [15].

Table 1. Error values of different architectures of NN with a variation of time delay

Characteristics of the NN	**Delay**	**Error**
Neural Network	0	1.7210^{-4}
RNN1	1	8.910^{-5}
RNN2	1	4.710^{-5}
RNN3	1	3.910^{-5}
RNN4	1	7.610^{-5}
RNN5	0	3.310^{-3}
RNN6	2	5.210^{-3}
RNN7	1	2.2410^{-4}
RNN8	1	1.510^{-4}
RNN9	1	2.2310^{-4}
RNN with e-kalman	**1**	1.8810^{-5}

Where: RNN1 : RNN with connections between **5** neurons in hidden layer.
RNN2 : RNN with connections between **8** neurons in hidden layer.
RNN3 : RNN with connections between **10** neurons in hidden layer.
RNN4 : RNN with connections between **12** neurons in hidden layer.
RNN5 : RNN with connections between **10** neurons in hidden layer where delay=0.

RNN6 : RNN with connections between **12** neurons in hidden layer where delay=2.
RNN7 : Elmann network with connections between **10** neurons in hidden layer and input layer.
RNN8 : Output recurrent network with connections between the output layer and input layer (10 neurons in hidden layer).
RNN9 : Recurrent neural network with connections between hidden (10 neurons) and input layers and between output and input layers.

3 Tests and Evaluation

Neural architecture considered in this investigation is a recurrent and multi-layer neural network. This architecture is composed of three layers arranged in a feed-forward model: The first layer is the input layer with a vector of 10 inputs, which are the joints of the lower trunk at instant t (two joints of each hip, one joint of each knee and two joints of each ankle). The second layer is the hidden layer where the number of neurons is fixed after many tests. The Third layer is the output vector that contains the 10 joint angles at instant $t+1$.

In order to evaluate the performance of each RNN, we calculate and compare the errors of many structures of the RNN. We start with the test of a simple NN. Then, we change the number of interconnected neurons in hidden layer (RNN1 to RNN6). After that, we add connections between neurons in different layers (RNN7 to RNN9). The RNN with e-kalman is, also, tested. It is a RNN for EKF-BPTT training. So, the filter of e-kalman are used in the BPTT algorithm and exactly in the step of update weights [5]. The goal of this evaluation is to compare several architectures of the RNN when a time delay is included in the dynamic equation.

So, table 1 gives a comparison between the error results of RNN with a variation of feedback connections between neurons and the different time delays (*delay=0, delay=1 and delay=2*). Results are summarized in table 1 and presented in figure 2 which clearly demonstrate the following:

1. The RNN with delay different to 1 have the worst performance. It reveals that wrong delay that occured at the beginning of each recursion will accumulate and propagate to the future when recursively, which will result in poor forecast accuracy.
2. RNN with ten neurons with connections in the hidden layer and a discrete delay equal to 1, gives a good performance.
3. the RNN mixed with e-kalman gives the best performance. It is clear that the last RNN is more important than the others. After computing, the error reached by the end of the training (600 iterations) is equal to 1.8810^{-5}. Figure 2 presents the desired right knee angle (in green), the estimated output of RNN with ten neurons with connections in the hidden layer (in red) and the output of the RNN mixed with e-kalman filter(in blue *). The two last curves are superposed because, as it is known, errors are in the order of 10^{-5}.

This architecture is described in figure 1 and characterized by one feedback connection which is a recursive connection between the neurons in the hidden layer. The numbers of the processing neurons and layers determined above are identified the best by a number of trials. In order to validate our RNN system, the output results are tested in a 3D simulator of the human-like robot. In each time step during the test phase, the Center

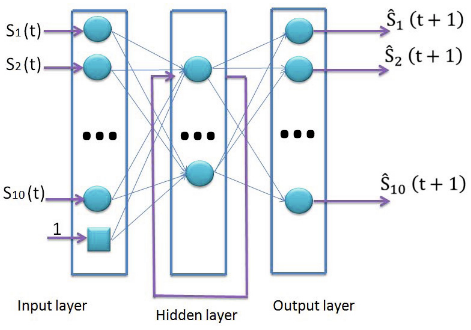


Fig. 1. The proposed RNN architecture

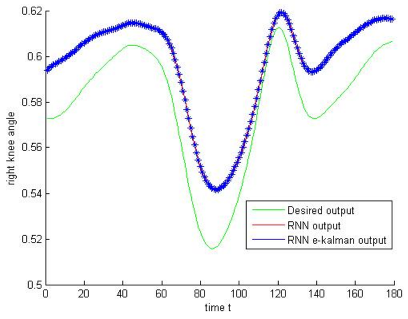


Fig. 2. Comparison between the desired output (green line), the RNN output (red line) and the output of RNN mixed with e-kalman (blue *)

Of Mass (COM) coordinates of the simulator are computed [13]. Then, the equations of stability of the simulator are calculated in order to avoid the risk of falling on the floor. When the simulator starts to walk, the stability can be controlled statically or dynamically. In both control manners, the COM of the simulator is computed.

Compared to [6] and [7], they predict positions of the center of mass of the biped. This prediction is controlled by the calculated ZMP stability. However, the COM coordinates information is not enough to control all the joints positions or angles.

In our case, the prediction of angles of articulations is controlled by the real data of the human subjects. The stability is controlled after the learning process.

Nakanishi et al report on their research for learning the biped locomotion from human demonstration. They choose to use database of walking joints of only one man from a book. Only the desired trajectories of the right leg are generated, while those of the left are concluded by shifting the phase of the oscillator of the right leg by π. The demonstrated trajectories are learned using locally weighted regression [11]. In the dynamical movement primitives, kinematic movement plans are described in a set of nonlinear differential equations with well-defined attractor dynamics. In our case, no differential equations of walking is used. The human motion is learned using RNN learning. The mentioned architecture of RNN using discrete delay is implemented with the technique of Euler discretization of the simplified continuous equation [1], the approximated equation is computed in the novel RNN architecture. The results of the proposed RNN are applied in 3D simulator.

The simulator is a geometric model similar to the human body. The 3D biped simulator has not only two legs and a pelvis but also a trunk, two hands and head. In our project, the higher part is immobile ; only the lower part is considered in the gait.

4 Conclusions

In this paper, the learning system of the human walking is proposed. This system is divided in two parts. The first is the building of the human walking model. A self made biomechanical data set of HBM walking is prepared. Only 10 joint angles at different instances are chosen from this data set. These joints are the inputs and the desired outputs of the RNN.

Different network architectures are tested. The training error is computed and the RNN with time delay and feedback connections in hidden layer and mixed with e-kalman has a good performance.

The second part is the prediction of the next joint angles in function of the actual joint angles. The role of the proposed RNN architecture is to obtain ”human-like robot behavior and predict the next values of the joint articulations”, but the RNN presented can only do prediction. To achieve control we, also, need other mechanisms (eg a PID or fuzzy controller on the joints). We mention that this system lacks the link between the learning step and the stability control.

As a future work, we attempt to expand the dynamic equation of the RNN by adding a continuous delay so as to manage the lateness of the network. We, also, aim at exploiting the described model to a Bidirectional Associative Memory (BAM) neural networks as a learning system in humanoid robotics. Another interesting topic for future work is by extending our approach with other databases of different motions (like: running, jumping, etc.). Furthermore, it is also planned to use this proposed system to learn elderly and handicapped persons.

Acknowledgment. The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

References

1. Ammar, B., Chérif, F., Alimi, A.M.: Existence and Uniqueness of Pseudo Almost-Periodic Solutions of Recurrent Neural Networks with Time-Varying Coefficients and Mixed Delays. IEEE Transactions on Neural Networks and Learning System 23(1) (2012)

2. Azevedo, C., Poignet, P., Espinau, B.: Artificial locomotion control: from human to robots. Robotics and Autonomous Systems 47, 203–204 (2004)
3. Baccour, N., Kaaniche, H., Chtourou, M., Ben Jemaa, M.: Recurrent Neural Network based time series prediction: Particular design problems. In: International Conference on Smart Systems and Devices, SSD (2007)
4. Bouaziz, S., Dhahri, H., Alimi, A.M.: Evolving Flexible Beta Operator Neural Trees (FBONT) for Time Series Forecasting. In: Huang, T., Zeng, Z., Li, C., Leung, C.S. (eds.) ICONIP 2012, Part III. LNCS, vol. 7665, pp. 17–24. Springer, Heidelberg (2012)
5. Bousnina, S., Ammar, B., Baklouti, N., Alimi, M.A.: Learning system for mobile robot detection and tracking. In: International Conference on Communications and Information Technology, ICCIT, pp. 384–389 (June 2012)
6. Concalves, J.B., Zampieri, D.E.: Recurrent neural network approaches for biped walking robot based on zero-moment point criterion. J. Braz. Soc. Mech. Sci. and Eng. 25(1), 69–78 (2003)
7. Erbatur, K., Kurt, O.: Natural ZMP Trajectories for Biped Robot Reference Generation. IEEE Transactions on Industrial Electronics 56(3), 835–845 (2009)
8. Huynh, T.Q., Reggia, J.A.: Symbolic Representation of Recurrent Neural Network Dynamics. IEEE Transactions on Neural Networks and Learning Systems 23(10), 1649–1658 (2012)
9. Manabe, Y., Chakraborty, B., Fujita, H.: Structural learning of multilayer feed forward neural networks for continuous valued functions. In: The 47th Midwest Symposium on Circuits and Systems, vol. 3, pp. 77–80 (2004)
10. Mandic, D., Chambers, J.: Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability (2001)
11. Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., Kawato, M.: Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems 47(23), 79–91 (2004)
12. Rokbani, N., Ammar, B., Alimi, A.M.: Toward intelligent biped-humanoids gaits generation. In: Humanoid Robots, pp. 259–271. I-Tech Education and Publishing, Vienna (2008)
13. Rokbani, N., Benbousaada, E., Ammar, B., Alimi, A.M.: Biped Robot Control Using Particle Swarm Optimization. In: IEEE International Conference on Systems Man and Cybernetics, SMC, pp. 506–512 (2010)
14. Stanton, C., Williams, M.-A.: Book Review: Robotics: State of the Art and Future Challenges. Artificial Intelligence (2008)
15. Welch, G., Bishop, G.: An introduction to the Kalman filter (1995)
16. Winter, D.A.: Biomechanics and motor control of human movement. John Wiley, USA (1990)
17. Wu, G.: Isb recommendation and definitions of joint coordinate system of various joints for the reporting of human joint motion. Part I: ankle, hip and Spine. Journal of Biomechanics 35 (2002)
18. Zhang, H., Liu, Z., Huang, G.: Novel Delay-Dependent Robust Stability Analysis for Switched Neutral-Type Neural Networks With Time-Varying Delays via SC Technique. IEEE Transactions on Systems, Man and Cybernetics -PART B: Cybernetics 40(6) (December 2010)

The Imbalance Network and Incremental Evolution for Mobile Robot Nervous System Design

Paul Olivier and Juan Manuel Moreno Arostegui

Universitat Politecnica de Catalunya, Department of Electronic Engineering, Campus Nord, Building C4, c/Jordi Girona 1-3, 08034, Barcelona, Spain
paul.olivier@upc.edu

Abstract. Automatic design of neurocontrollers (as in Evoluationary Robotics) utilizes incremental evolution to solve for more complex behaviors. Also manual design techniques such as task decomposition are employed. Manual design itself can benefit from focusing on using incremental evolution to add more automatic design. The imbalance network is a neural network that integrates incremental evolution with an incremental design process without the need for task decomposition. Instead, the imbalance network uses the mechanism of the equilibrium-action cycle to structure the network while emphasizing behavior emergence. An example 11-step design (including a 5-step evolutionary process) is briefly mentioned to help ground the imbalance network concepts.

Keywords: imbalance network, equilibrium-action cycle, incremental evolution, emergence, task decomposition.

1 Introduction

Neuro-evolution, in particular, evolutionary robotics (ER), has no doubt shown the power of artificial evolution in the design of robotic behavioral controllers [1]. In particular, it offers a promising alternative to manual design in which each network parameter (connection, weight, threshold, and element type) must be clearly understood in terms of its reason for being present in the network and its particular role in one or more behaviors. ER utilizes the artificial neural network (ANN) as basis for designing a neurocontroller (a popular term for an evolved ANN-based nervous system for determining a robotic agent's behavior) and can be a single ANN or consist of several subnetworks [2]. Being based on ANN's, the neurocontroller exhibits robustness to noise as well as an inherent learning capacity that is promising when designing for an uncertain dynamic environment in which sufficient modeling of the robot and/or the environment is not viable. As particular advantage over manual design, the designer is freed from the architectural details of how to define neuronal elements and interconnect them since a suitable architecture can be automatically evolved by specifying minimal details (an initial network topology (the topology itself can be evolved [3]), the neuron model, input/output connections) [4]. Since ER uses an evolutionary algorithm the conventional learning algorithm is replaced by a fitness function, which

© Springer-Verlag Berlin Heidelberg 2013

includes factors such as the environment and the robot's morphology. As an additional property, in the final neurocontroller each behavior's implementation is spread across several nodes, which provides robustness against lesions and sensor inaccuracies [5]. However, ER research has for many years focused only on single-network neurocontrollers and simple behaviors. Moving to more complex behavior (such as wall-following, obstacle avoidance and predator-prey scenarios) has identified several issues such as increased difficulty in fitness function design, and bootstrapping (the evolutionary process cannot start because all individuals perform equally poorly) [2][6]. This has lead to incremental evolution in which there are several stages of evolution across which increased levels of difficulty are applied, for example, to the desired behavior and the environmental complexity [6]. A complementary approach is task decomposition by which the desired behavior is broken down into simpler behavioral modules, with a separate subnetwork evolved (in a chaining or parallel way) for each module as additional incremental steps [2]. As promising as the field of ER appears, it is important to see automatic and manual design as complementary instead of opposing approaches. For example, more manual design such as task decomposition is required as more complex behavior is being evolved. That said, some do advocate using automatic methods for task decomposition [6]. A greater problem with task decomposition is that it is a long-existing concept in manual design approaches such as behavior-based robotics (in particular, the subsumption architecture) with reported problems of scalability [4]. Furthermore, it is not always clear how to do this breakdown and layering of behavior from complex to simple. Instead of task decomposition, of greater importance is to identify a behavior-causing mechanism: The mapping between neural mechanisms and the observed behavior is not as direct as implied by task decomposition [7].

Due to behaviors being spread across many nodes of the ANN, and the unclear link between a network parameter and one or more behaviors, the ER solution is susceptible to change. Here, "change" refers to a manual change by the designer (adding or adjusting some network or algorithmic parameter to add new behavior or fix/enhance existing behavior, followed by recompiling, programming and execution) as opposed to a change resulting during the automatic design process. If in the end all the connection, node and weight decisions are included in the automatic process, then it will be the setup/design of the automatic process that will experience these manual changes. Where an experiment mostly consists of one to several design steps (that is, manual changes), for a complete robot with a rich behavior repertoire it is better to imagine hundreds of steps. Even if design is modularized and distributed across various design teams, the final product is due to a large number of steps involving assembly, integration, improvements and bug fixing. Since automatic design uses more manual effort in solving for complex behavior, so should manual design maximize any form of automatic design. One way to do this is to use artificial evolution. There have been attempts in the past to use evolution with manual design by replacing decomposed low-level behavior modules with small ANN's, or evolving a high-level controller acting upon hand-designed low-level behavior modules [8]. The approach in this article is to identify "uncertain" network parameters whose values are difficult to determine a priori due to a lack of robot/environment models or due to environment complexity. The evolution process performs more adaptation rather than solving an optimization problem, which is a focus also advocated for ER [9]. In addition to increased automation, manual design should avoid overspecifying the implementation of a behavior (a consequence of industrial robotics' controlled environment in which every motion is designed). This requires manual design to maximize emergence, which can be defined as novel high-level behavioral phenomena resulting from low-level behavioral modules. Emergence requires focusing on the details of what must be done instead of how to perform every motion of every behavior.

The remainder of this article discusses the imbalance network in terms of dealing with the issues and challenges raised above for manual design. Section 2 reviews the equilibrium-action cycle. Section 3 uses an example imbalance network design to elucidate some of the key aspects, focusing on incremental evolution in the imbalance network's design process. Section 4 gives a brief conclusion.

2 The Imbalance Network and Equilibrium-Action Cycle

The imbalance network is a dynamic parallel biologically-inspired neural network sharing features with biological neural networks, dynamic recurrent neural networks and behavior-based robotics [4]. The network is constructed using only neuron-like nodes called imbalance elements. In spite of the features shared, an imbalance element is not modeled after a neuron but after the concept of the equilibrium-action cycle (EAC) [10]. The EAC plays the most fundamental role of all the imbalance network principles in guiding the design of the imbalance network to be conducive to change and evolutionary process integration. The EAC states that all action results from cyclic neural processes and that the purpose of this action is particular to the reestablishment of equilibrium. Particular stimuli (their presence or absence) maintain the state of equilibrium and therefore to return to equilibrium the action has to reestablish these equilibrium-causing stimuli (see Fig. 1). The EAC is completely generic: Stimuli do not always refer to sensory input and are also internal signals generated within the nervous system. Action is not always observable motion but could mean "activation", "filtered information" or "useful event/state". The EAC, together with inspiration from the neuron, was used to develop a computational element, called the imbalance element, with which the complete imbalance network is constructed. The output of the imbalance element is its imbalance state, which is connected to either a motor output or an input of another element. The EAC applies to behavior at all levels: from low- (reflexes) to high-level (the organism interacting as a motivated entity upon its environment). This means all observed behavior results from these cycles but with the distinction that the primary sense of each behavior lies in its role to reestablish equilibrium in one or more imbalance elements. The imbalance network has a single function which is to try and maintain balance from element level to network level. It is in the strive for equilibrium that the imbalance network exhibits the necessary dynamics for reaching and maintaining equilibrium between the robot and its environment, a feat only achieved by a fit individual. The natural incorporation of evolution into the design process stems from this link between a fit individual and an imbalance network sufficiently able to maintain equilibrium during which the desired behavior is achieved. From the designer's point of view the ultimate goal is for the robot to show the desired behavior, that is, to perform some useful task. Yet, the design process concerns the definition and integration of EAC's, not only for yielding the desired observable behavior but also the "actions" (from perception to coordinating motor output) inside the imbalance network. In this indirect sense of behavior design, it might appear simpler to apply conventional manual design. Via an EAC focus the designer is freed from defining a priori particular behavior layers, topologies or hierarchical structures, or figuring out the right task decomposition. The EAC is a sufficient unified mechanism that inherently emphasizes emergence due to the indirect way of specifying behavior (behavior is not specified per se but by its role in an EAC). Fig. 1 shows two minimal one-element imbalance networks for a mobile robot with a distance sensor at the front, highlighting the two basic modes of the imbalance element: imbalance due to the presence of a stimulus/object (left), and due to the absence of a stimulus/object (right). Each imbalance consists of a value, with excitatory/inhibitory input ports involved in increasing/decreasing the value, and with an imbalance state and thresholds to determine its output. The value increases at a rate determined by the total drive and rate-increase input and, whenever the total drive input is zero, decreases at a rate determined by the total rate-decrease input. In each case the robot is initially standing still because the imbalance element is in balance.

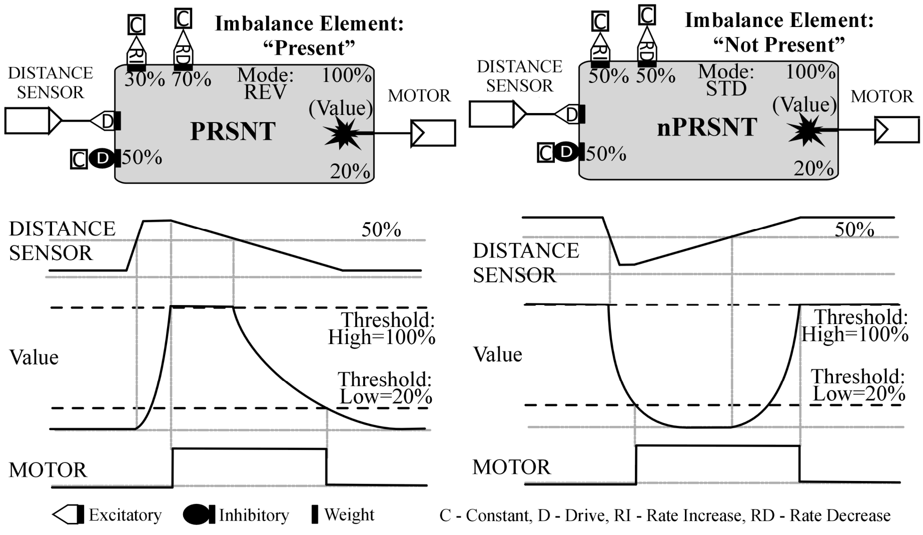


Fig. 1. Two one-element imbalance network examples. An object is moved closer to (left) or further from (right) the distance sensor, causing imbalance (motor activation) and movement to restore balance by moving away from (left) or closer to (right) the object. See text for details.

3 Example Design: Moving Down a Tunnel

To help elucidate the imbalance network, let us look at some aspects of an example imbalance network design which simply has to get a robot to move down a narrow tunnel without touching the walls. The robot used is the small, circular e-puck (16-bit dsPIC microcontroller @ 40 MIPS, a ring of 8 infrared distance sensors (of which only the left, right and rear left sensors are used in this design) and LED's, two stepper motors) [11]. The design consists of 11 steps, with evolution featuring in steps 1, 2, 9, 10 and 11. This means that evolution is inherently incremental since it forms part of the incremental design process. The final design is shown in Fig. 2. Note that this is the complete design instead of a high-level or functional description. This exact image (the ports, elements, connections, weights and thresholds) is described in a script file, from which the C code to execute on the robot is automatically built. Designing the imbalance network involves no conventional programming. Each imbalance element's output is updated in parallel every 2ms (the network update cycle). The design process for each step involves firstly using simulation (which could include using sensory data sampled from the robot) for the functional design of new or modified elements. This will be followed by trial runs on the real robot to confirm correct behavior. Whenever an "uncertain" parameter is defined, the design step will include adaptation of this parameter using the evolutionary process (see further below). This parameter can be seeded with good estimates obtained from simulation, or with values that span the complete parameter's value range. Due to space limitations, instead of a full description, the comments below describe the issues discussed previously.

As an example of emergence, note that there are no turning or forward movement modules. These motions are the result of the EAC's designed for the Left Pressure (L_PRSSR) and Right Pressure (R_PRSSR) elements. Left Pressure becomes unbalanced if the distance to the left wall (measured by the left sensor) is less than a threshold, which is set by the inhibitory drive input. One action that will cause the element to return to balance (that is, measure a distance more than the threshold) is to turn the left wheel, thereby turning the robot away from the left wall. Thus, the Left Pressure element's output is connected to drive the Left Motor element (and hence the left wheel) such that the left wheel will turn while Left Pressure is imbalanced. If we add a complementary right side mechanism (the Right Pressure and Right Motor elements), and given we place the robot in the middle of a tunnel with walls just close enough for both Left and Right Pressure elements to be imbalanced, then the emerging behavior is forward movement of the robot down the tunnel with corrective turning motion when the robot gets too close to any of the walls.

The detection threshold (realized by the inhibitory drive input's weight) of Left Pressure and Right Pressure are considered uncertain because of the need to take into account sensor inaccuracies, ambient light conditions, robot morphology, environmental characteristics and forward speed. Therefore, these two weights are evolved. The evolutionary mechanism (selection, mutation, fitness evaluation) is completely online, designed to use few memory and processing resources, based on a variable with a gaussian distribution for mutations for asexual reproduction (no crossover), with survival based on a true/false decision made either by an observer or programmed automatically into the code (that will expect a particular condition to serve as true, that is, "survived") [10]. Nonuniform mutation is realized by way of a mutation weight that changes according to the number of survivors per generation. A larger mutation weight means that mutational changes will be larger, and for a smaller mutation weight that the changes will be smaller. For this design the gaussian variable is a 6-bit signed value and the maximum mutation weight is 6 (meaning the mutational change will be the gaussian variable shifted by 6 bits towards the most-significant bit). If no individual survived, then the mutation weight is increased by two, else it is decreased by two. The population consisted of 16 individuals.

The evolutionary weight (distance threshold) for Left Pressure is evolved in step 1 (evolution step A), with the individuals seeded with values to cover the complete input range. Step 2 (evolution step B) involves evolution of the evolutionary weight for Right Pressure. However, its individuals are seeded with the results from step 1. During this step, evolution is disabled for Left Pressure. Step 3 adds the Left Motor and Right Motor elements. In step 4 to 6 the corrective action's details are designed, which includes elements L_SPACE, L_SHCK_DET, L_SHCK_DLY, the complementary right side elements, and LR_SHCK_DLY_RDEC. The strategy is to detect the moment when any side stops detecting its wall, and then apply a corrective action for a specific duration or until both walls are detected. The duration is fixed by the excitatory drive input's weight of LR_SHCK_DLY_RDEC, which is considered the third "uncertain" parameter and that is evolved during step 9 (evolution step C). The individuals are initially assigned values that span the full range. During this step evolution is disabled for the previously evolved weights (steps 1 and 2). In addition, element LR_PRSSR_MEM halts the robot after exiting the tunnel successfully (step 7) and element DAMP ensures that the start of any movement by the robot after power-up requires manual indication by the user (step 8), which is to touch the left rear sensor. In step 10 (evolution step D) evolution is enabled for all three evolved weights as a final stage of improving equilibrium between the robot and its environment. In step 11 (evolution step E) the environment is changed abruptly to test the current solution's adaptive capacity.

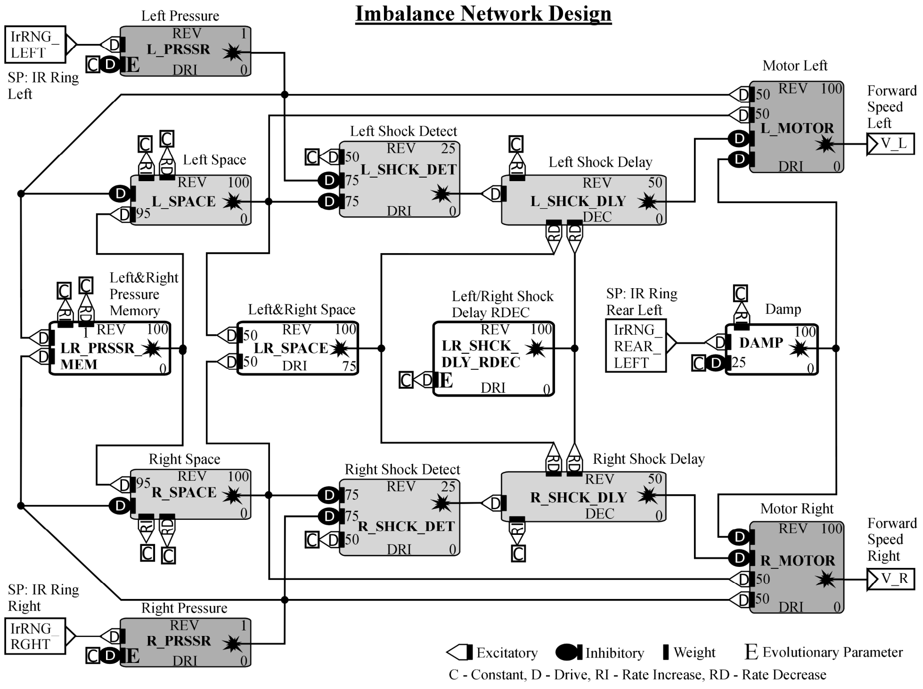


Fig. 2. The complete imbalance network design for moving down a tunnel

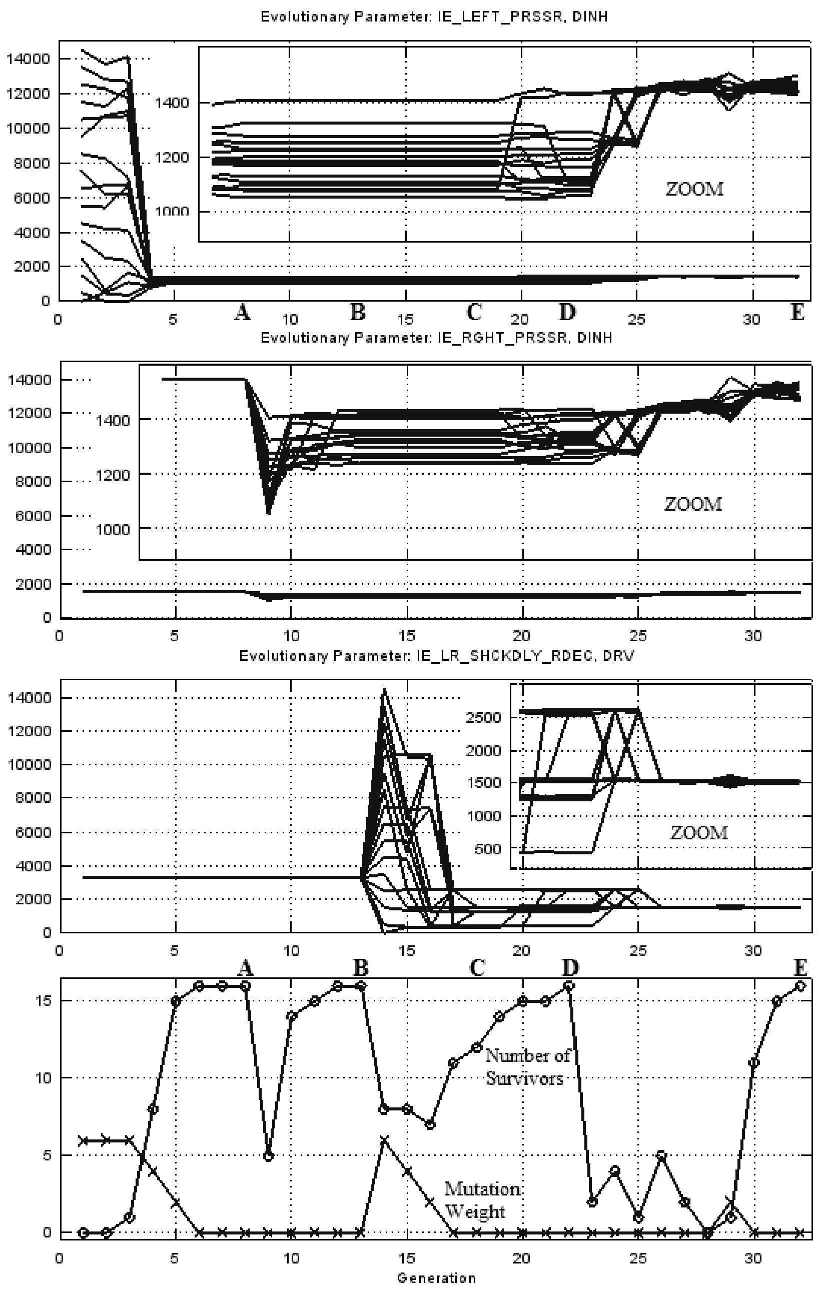


Fig. 3. The results from the evolution steps A to E

As environmental change the tunnel width is reduced from 14,5cm to 11,5cm. The results from the five evolution steps (A to E) are shown in Fig. 3. Together, all steps only required 32 generations. The more restrictive environment of step E has decreased the solution space of each evolutionary parameter, which is in line with results from incremental evolution in ER [5]. The Left/Right Pressure evolutionary weights show an increase (the wall detection distance is shorter), as can be expected from the reduced tunnel width. From this example it is clear that the design is a process of incremental network composition, building towards the desired behavior without any clear task decomposition mapping and behavioral levels/hierarchy.

4 Conclusion

The imbalance network represents an alternative manual design methodology that naturally integrates design-for-emergence and evolutionary stages in an incremental design process. A brief description of an example design (a robot moving down a tunnel) has tried to ground the main concepts, in particular the fundamental role of the equilibrium-action cycle as a mechanism for structuring the imbalance network without task decomposition and its associated mapping of behavior to network modules.

References

1. Nelson, A.L., Barlow, G.J., Doitsidis, L.: Fitness functions in evolutionary robotics: A survey and analysis. Robotics and Autonomous Systems 57(4), 345–370 (2009)
2. Duarte, M., Oliveira, S., Christensen, A.L.: Hierarchical evolution of robotic controllers for complex tasks. In: 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL), pp. 1–6. IEEE (November 2012)
3. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence 1(1), 47–62 (2008)
4. Harvey, I., Husbands, P., Cliff, D., Thompson, A., Jakobi, N.: Evolutionary robotics: the Sussex approach. Robotics and Autonomous Systems 20(2), 205–224 (1997)
5. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior 5(3-4), 317–342 (1997)
6. Mouret, J.B., Doncieux, S.: Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 1161–1168. IEEE (May 2009)
7. Seth, A.K.: Evolving action selection and selective attention without actions, attention, or selection. In: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior on From Animals to Animats, vol. 5, pp. 139–146 (September 1998)
8. Togelius, J.: Evolution of a subsumption architecture neurocontroller. Journal of Intelligent and Fuzzy Systems 15(1), 15–20 (2004)
9. Harvey, I.: Cognition is not computation; evolution is not optimisation. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 685–690. Springer, Heidelberg (1997)
10. Olivier, P., Arostegui, J.M.M.: The Equilibrium-action cycle as a mechanism for design-evolution integration in autonomous behavior design. In: 2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 190–197. IEEE (June 2012)
11. E-puck educational robot, `http://www.e-puck.org/`

Balancing of a Simulated Inverted Pendulum Using the NeuraBase Network Model

Robert Hercus, Kit-Yee Wong, and Kim-Fong Ho

Neuramatix Sdn Bhd, Kuala Lumpur, 59200 Malaysia
{Hercus,KitYee,kfho}@neuramatix.com

Abstract. This paper presents an alternative approach for the control and balancing operations of a simulated inverted pendulum. The proposed method uses a neuronal network called NeuraBase to learn the sensor events obtained via a simulated rotary encoder and a simulated stepper motor, which rotates the swinging arm. A neuron layer called the controller network will link the sensor neuron events to the motor neurons. The proposed NeuraBase network model (NNM) has demonstrated its ability to successfully control the balancing operation of the pendulum, in the absence of a dynamic model and theoretical control methods.

Keywords: Neural Network, Inverted Pendulum.

1 Introduction

The inverted pendulum is a classic problem in non-linear control. The dynamics of an inverted pendulum form the basis of many diverse phenomena such as walking, aircraft roll control and planar robot arm control. The inverted pendulum has been widely used as a platform for testing the efficacy of various types of controllers [1-12]. However, the controller design of an inverted pendulum is difficult due to its multi-variability and inherent instability.

There are, generally, two categories of controllers namely dynamic modelling and control and machine learning. In dynamic modelling and control, controllers using the methods of Proportional-Integral-Derivative (PID), pole placement, Linear-Quadratic-Regulator (LQR) and Linear-Quadratic-Gaussian (LQG), have been previously applied to the control of inverted pendulum systems [1-4]. Although most of these approaches achieved good balancing performance, an accurate mathematical representation for the inverted pendulum is not easily formulated as extensive knowledge of the system dynamics is required. Machine learning approaches such as fuzzy logic, artificial neural networks, neuro-fuzzy, self-organizing maps, recurrent neural networks, cerebellar model articulation controller (CMAC) and genetic algorithms have also been studied extensively [5-12]. The capabilities of these machine learning methods in mapping non-linearity and in dealing with uncertainties in system parameters, eliminate the need for exact mathematical models.

In this paper, the balancing control of a simulated inverted pendulum using a temporal-based neural network model named NeuraBase [13] is presented.

© Springer-Verlag Berlin Heidelberg 2013

This implementation is not intended to make direct comparisons with other pendulum balancing methods, but to introduce an alternative and novel approach.

The NeuraBase generic toolbox can be downloaded at [14]. The dynamics of the simulated inverted pendulum are adapted from [15]. The swing-up process necessary to bring the pendulum to the balancing region of 0±10° at the outset was controlled using a separate NeuraBase controller, and will not be discussed in this paper. As shown in Figure 1, the rotary inverted pendulum consists of a pivot arm rotating in a horizontal plane by means of a motor. At the other end of the arm, a pendulum is mounted, rotating in a plane that is always perpendicular to the rotating arm.

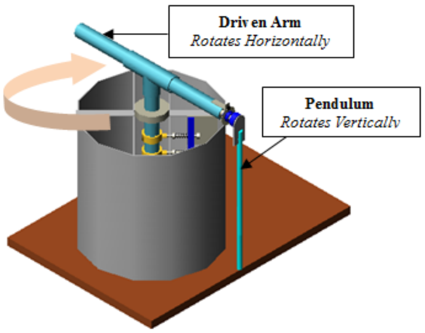


Fig. 1. Drawing of the rotary inverted pendulum

This article is organized as follows. Section 2 of this paper describes the usage of the NeuraBase Network Model (NNM) as a controller. Section 3 gives the simulation set-up. Section 4 describes the learning logic used with the NNM, and section 5 presents the results and discussion.

2 NeuraBase Network Model (NNM)

The NNM is a network data structure that can store a sequence of events. As shown in Figure 2 below, the neurons in a NNM can be associated temporally or spatially. The basic unit is a neuron. Each neuron represents an event. Two neurons can be joined to represent a sequence of sensor or motor events. The way events are constructed in the NNM provides for fast searching and matching.

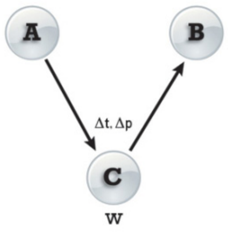


Fig. 2. The NeuraBase Network Model where *t* denotes time proximity and *p* denotes spatial proximity

The proposed NNM controller for the inverted pendulum consists of three different networks as shown in Figure 3. For the inverted pendulum, both the motor and controller networks have a single level architecture and the sensor network has a multilevel architecture.

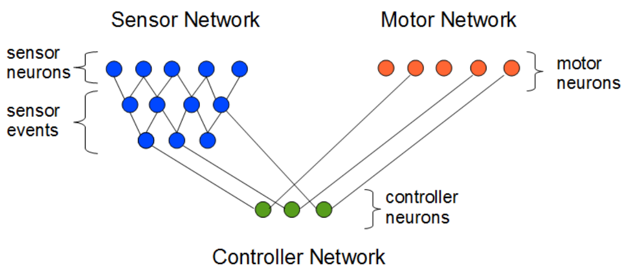


Fig. 3. The architecture of the network of NeuraBase used in the balancing of the inverted pendulum

These three networks store different types of events, namely a) sensor neurons and events - input to the system (the pendulum position readout from the encoder); b) motor neurons - outputs from the system (the motor velocity change for driving the arm); c) controller neurons - associations between the sensor events and motor neurons. Each type of event builds up an association of events in their respective network. The sensor network, motor network and the controller network store sensor neurons and events, motor neurons and events, and controller neurons respectively. The simplified data structure of neurons used in NNM is described in Table 1. Each neuron has a memory capacity of 40 bytes but not every one of the fields is necessary for the inverted pendulum problem. More detailed descriptions of the sensor, motor and controller neurons are provided in Section 3.

Table 1. Data structure of a neuron (simplified), * denotes fields only applicable to the controller neuron

Field	Data Type
Head	unsigned int
Tail	unsigned int
Successor	unsigned int
Frequency/ Weight*	signed int
Next	unsigned int
Overshoot/Undershoot Flags*	unsigned short

3 Simulation Setup

The NNM was used as the control system for the inverted pendulum. In contrast to other control methods [1-4] which require knowledge of all four state variables i.e. the pendulum position and velocity, the arm position and velocity, the NNM only needs two variables, namely the pendulum position and the change in arm velocity.

To further simplify the model, a segmentation scheme was used to represent the defined range of angles on the pendulum positions. As shown in Figure 4 below, the segments are defined by the pendulum's position with reference to the upright (0°) position. The operating region for the balancing mode was thus set at ±10° from the upright position of 0°. The angular measurement was divided into two symmetrical sides, whereby, the right side was represented by positive segments and the left side was represented by negative segments. There are, altogether, 21 segments; hence, there are only 21 basic sensor neurons underlying a sequence of sensor events for the inverted pendulum balancing model.

For motor neurons, the stepper motor rotation was controlled by changing the velocity between two time samples. Each motor neuron ***M*** was represented as a unit velocity change which is then applied to the arm rotation, where each unit corresponded to a shift of 0.02πrad/s, with either a positive or negative sign representing counter-clockwise or clockwise rotation, respectively. With a change limit of πrad/s imposed in either direction (±*M50*), there were consequently 101 basic motor neurons defined to represent these changes in arm velocity.

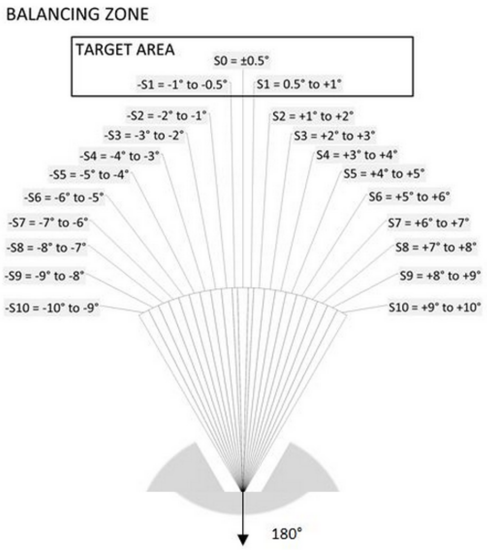


Fig. 4. The segmentation of angular positions of the inverted pendulum in the balancing zone

The controller neurons associate sensor events with motor neurons. The association indicates the possible variables to control the pendulum through a learning process. The inverted pendulum problem is proposed to be solved using the learning model - given a sequence of angular positions (***S***) of the pendulum, a change in motor speed or arm velocity (***M***) can be applied to move the pendulum to the desired balance position. The relationship between the sensor events and the motor neurons are associated (linked) via a controller neuron. At each 13ms interval, ***S*** was obtained and stored in the sensor network, forming a sequence of position segments. Similarly, each ***M*** defined for the motor was stored in the motor network.

Figure 5a depicts the neuron structure of a sample sensor event *C* within the sensor network representing the sequence of positions *C* = {S6, S4, S1}. The neuron *A* (head of *C*) represents the sensor sequence of {S6, S4} and the neuron *B* (tail of *C*) represents the sensor sequence of {S4, S1}. Neuron *D* represents a set of successors of *C* which can be a sequence of position segments {S6, S4, S1, S0} or *D* could be controller neurons for *C* linking to a specific motor action.

Each motor neuron represents a unit velocity change to the arm rotation. Figure 5b depicts the neuron structure of a sample controller neuron (*E*) within the controller network. The neuron *C* (head of *E*) is the sensor event mentioned above and the neuron *F* (tail of *E*) is a motor neuron.

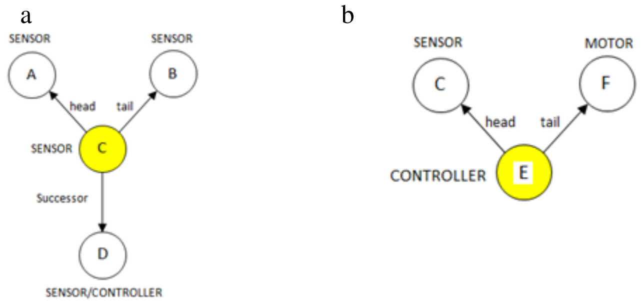


Fig. 5. a) The structure of a sensor event; b) The structure of a controller neuron

4 Learning Logic

As mentioned in the previous section, the operating region of the NNM controller was 0±10° or [-*S10*, *S10*], beyond which, the pendulum is deemed uncontrollable and control will be transferred over to the swing-up controller. Using the reinforcement learning method, the goal of the NNM controller is to bring the pendulum into the target region 0±1° or [-*S1*, *S1*] and maintain the pendulum in the target region for as long as possible.

The direction of motor speed change applied in order to balance the pendulum was based on the following learning rule: whenever the inverted pendulum falls out of the target [-*S1*, *S1*] region, the NNM controller will attempt to predict the best motor action to execute, by searching within the controller network for the strongest motor actions exceeding a predefined weight, which will bring the pendulum into the target area [-*S1, S1*]. The sensor sequence events have a fixed maximum length of *n*, which was set to 5. For instance, given a sequence of sensor events {-S3, -S2, -S2, -S1, -S1}, there are three learned candidate controller neurons N3, N4, N5 which correspond to motor neurons -2, -1, and 1, respectively. The strength of each controller neuron is represented by its trained weight; hence, the controller neuron with the highest weight will be the most reliable prediction because it is the accumulated result of learning, using both positive and negative feedbacks from past trials.

The association of the sensor event (***S***) and motor neuron (***M***) in the controller network is a reinforced learning process, whereby positive and negative feedbacks dictate how the learning takes place by tuning the weight of the controller neurons.

Based on a predefined learning goal, the positive and negative feedback rules were defined based on evidence to determine whether the goal was achieved. The feedback rule is constructed according to the strategy that, given a sequence of sensor events collected using a fixed time interval (e.g. pendulum traversal path from *S3* → *S1* → *S0* → *-S1* → *-S1*) and a motor neuron effecting a change in speed (e.g. *M3*) has been applied, the current sensor data is *-S1*. This represents the segment position of the pendulum after the speed change (*M3*) has been applied.

A positive feedback is given if the pendulum managed to reach within the target region of [-*S1*, *S1*], upon which, the weight of the controller neuron is incremented by 1. The more the controller neuron experiences positive feedbacks for its motor predictions, the stronger the link coupling will be, thereby resulting in a stronger positive memory of the respective motor action. A negative feedback is evoked if the pendulum fails to reach the target region [-*S1*, *S1*], upon which, the weight of the controller neuron is decremented by 1, thus reducing the coupling strength of that link. Alternatively, the NNM controller will create a controller neuron linking the sensor event to the motor neuron. Eventually, a network of almost all possible sensor events associated with motor actions will be stored within NeuraBase, and the controller neurons linking sensor events to the right motor actions will have higher weights compared to those linking sensor events to incorrect motor actions.

In the case where there are no controller neurons with strength exceeding the weight, which is common at the beginning of training, the controller will set the sensor sequence to its tail (meaning checking recursively for the latest *n-1, n-2,... 1* events for any matches). If none can be found even for the shortest sensor event, a random number within a set range will be picked.

This motor range is bounded by motor neurons linked to previously used controller neurons which have either been flagged as overshooting and/or undershooting controller neurons according to the rules outlined in Table 2. Each time a controller neuron's overshoot/undershoot flag is set, the set range becomes smaller, with the controller neuron's linked motor neuron as the new upper or lower boundary of the set. This method helps the controller narrow down its choices of approximately correct motor actions for the sensor event more quickly, compared to the controller having to attempt all motor neurons before finding a suitable one.

Table 2. Controller neuron overshoot/undershoot flagging rules

Pendulum position		Flag
Before: [-S10,-S2]	After: [S2,S10]	Overshoot
Before: [S2, S10]	After: [-S10,-S2]	Overshoot
Before: [-S10,-S2]	After: [-S10,-S2]	Undershoot
Before: [-S10,-S2]	After: [-S10,-S2]	Undershoot
Before: [-S10,-S2]	After: [-S1,S1]	None
Before: [S2,S10]	After: [-S1,S1]	None

5 Results and Discussions

The experiment's conditions were defined when the inverted pendulum was trained starting with a null NNM. In the first experiment, we fixed the initial position of the pendulum at *–S4* within the balancing mode operating region, that is, the pendulum is falling out of *–S4*.

5.1 Learning from a Fixed Initial Position

Training was truncated when the balanced duration exceeded a certain period. This is because in a simulation environment, the system dynamics are not expected to change much and the pendulum should balance indefinitely. The sample experiment shown in Figure 6, depicts the pendulum's average balancing time versus the number of trials). The truncation point was set at 2 minutes. Figure 6 also shows the growth trend of the neurons in the NNM, which correlates with the upward balancing trend, as shown in the same figure. Initially, the motor actions are all random values, so the balancing durations were generally less than 10 seconds. This was expected as the pendulum's stability does not only depend on the immediate motor action but also on all past motor actions, all of which, have not been learned extensively. After approximately 30 trials, the results showed that the NNM controller was able to balance the pendulum up to the truncation point. The controller is able to learn relatively quickly i.e. in 30 trials primarily due to the shrinking bounded range method introduced in Section 4, whereby the controller narrows down to a correct motor action based on knowledge of the existing controller neuron overshoot/undershoot flags. Also, this apparent sudden jump in performance can be explained by the fact that, once the NNM controller succeeded in bringing the pendulum into the target region and maintaining it there for a few time steps, the controller was able to sustain its balance, as the pendulum can be considered stable at that point.

At the commencement of training, the NNM did not contain any stored pattern, hence, many of the new patterns (new neurons) were created in the initial trials. The growth trend of NNM neurons approached saturation point after approximately 30 trials starting from position *–S4*. Once a good control pattern was achieved, it was able to repeat the same actions consistently, which led to a saturation point in both the balancing duration and neuron growth, as shown in Figure 6.

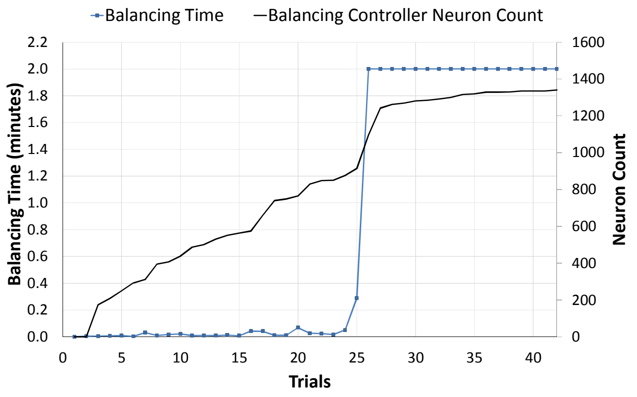
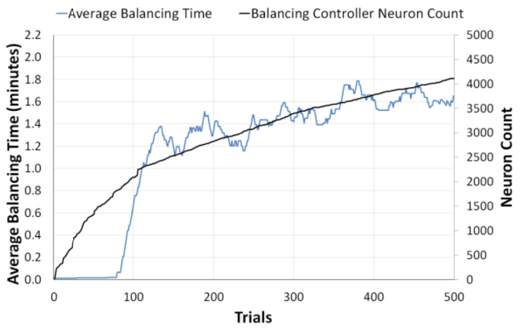


Fig. 6. Balancing time (in blue) and neuron count (in black) vs. number of trials

5.2 Learning from Random Initial Positions

As a form of variance, various different initial angles starting from [-10°, 10°] were chosen randomly. Figure 7a shows the average balancing time of a pendulum trained to balance in 500 trials using different initial angles. Due to the variance in the initial start condition, the controller took approximately 100 trials before it managed to register a first successful balancing trial. It is observed that the average balancing time was still about 20% lower than the balancing time truncation point (set at 2 minutes) after 500 trials, but it continued to improve with time as the NNM controller was still learning to cope with different starting conditions. The same reasoning is applied to the observation of the continuous growth of neurons.

a



b

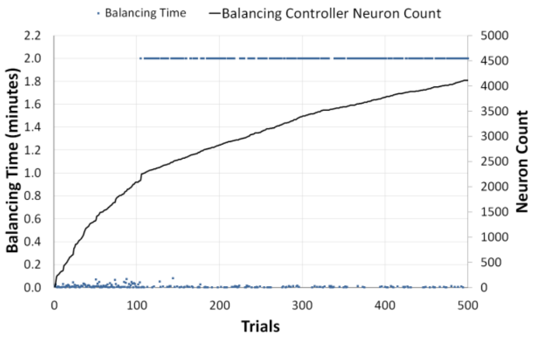
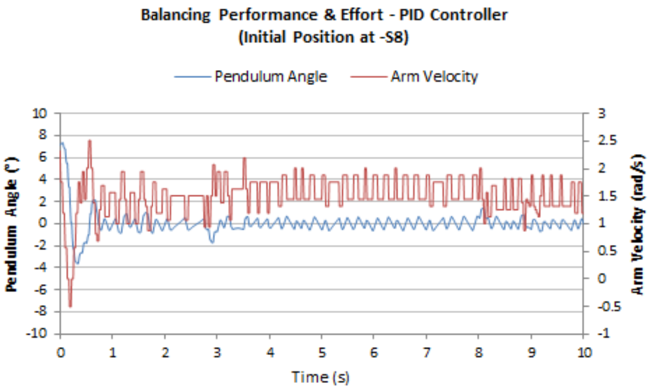


Fig. 7. a) Average balancing times (blue line) and neuron count (black line) vs. number of trials and; b) Actual balancing times per trial (blue scattered points) and neuron count (black line) vs. number of trials for the NNM controller trained using an initially empty balancing NeuraBase and starting from different random initial positions between [-10°, 10°]

5.3 Comparison with PID

Furthermore, we compared the balancing performance of NNM with the classical PID control method using the same initial pendulum segment of *–S8*. Figure 8a and 8b illustrate the performance comparison between the two said methods. The PID method shows oscillatory but saturating pendulum angles from the *–S8* segment to the target region of [-*S1*, *S1*] (segmentation was applied to the PID method and hence the desired target of PID was set at the region of [-*S1*, *S1*]), but PID is not a self-learning method as its parameters were tuned manually to achieve the optimal performance. On the other hand, the NNM does not require manual tuning and it is able to balance the pendulum after self-learning. Referring to Figure 8b, the alternating performance surrounding the S0 segment is the result of setting the balancing zone between the segments [-*S1*, *S1*] or [-179°, 179°]. In terms of control effort, the arm velocity needed to balance the pendulum must be maintained between [-0.1, 0.5] rad/sec. Using the PID method, the control effort must be maintained between [1.1, 1.8] rad/sec in order to maintain the balance. As can be seen, both models have very similar performance characteristics.

a



b

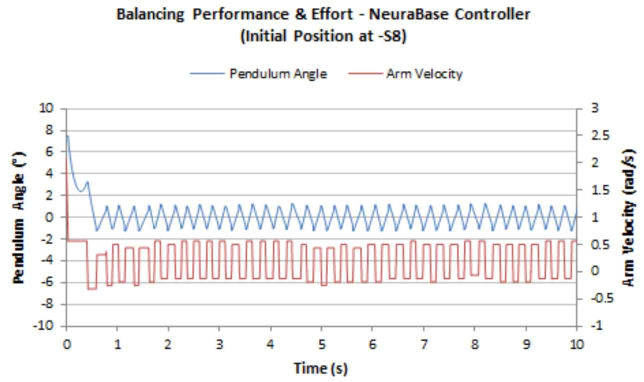


Fig. 8. a) The balancing performance and control effort of inverted pendulum using the PID controller; b) The balancing performance and control effort of inverted pendulum using the NNM controller

The NNM has unique advantages in: simple computations, inexpensive computational complexity, fast learning and the application of the NNM is intuitive and easy to understand.

6 Conclusion

The self-learning NNM controller presented in this paper is a proof of concept that the NNM can be easily adapted to handle the classic control problem for the balancing operation of a simulated inverted pendulum. The experimental results show that the NNM controller is comparable with the PID controller, however the former requires lower control effort than the latter.

For future work, the same control model will be applied to a hardware model of the inverted pendulum as well as control problems of higher complexity which involve more parameters, such as, the self-balancing of a bipedal walking robot.

References

1. Mertl, J., Sobota, J., Schlegel, M., Badal, P.: Swing-up and Stabilization of Rotary Inverted Pendulum. In: Proceedings of Process Control, pp. 1–6. Slovak University of Technology (2005)
2. Sukontanakarn, V., Parnichkun, M.: Real-time Optimal Control for Rotary Inverted Pendulum. American Journal of Applied Sciences 6(6), 1106–1115 (2009)
3. Astrom, K.J., Furuta, K.: Swinging up a Pendulum by Energy Control. Automatica 36(2), 287–295 (2000)
4. Nasir, A.N.K., Ahmad, M.A., Rahmat, M.F.: Performance Comparison between LQR and PID Controller for an Inverted Pendulum System. In: Proceedings of International Conference on Power Control and Optimization (2008)
5. Radhamohan, S.V., Subramaniam, M., Nigam, M.J.: Fuzzy Swing-up and Stabilization of Real Inverted Pendulum using Single Rulebase. Journal of Theoretical and Applied Information Technology, 43–50 (2005)
6. Minnaert, E., Hemmelman, B., Dolan, D.: Inverted Pendulum Design with Hardware Fuzzy Logic Controller. Journal of Systemics, Cybernetics and Informatics 6(3), 34–39 (2008)
7. Hayashi, I., Nomura, H., Wakami, N.: Acquisition of inference rules by neural network driven fuzzy reasoning. Japanese Journal of Fuzzy Theory and Systems 2(4), 453–469 (1990)
8. Zheng, Y., Luo, S., Lv, Z.: Control Double Inverted Pendulum by Reinforcement Learning with Double CMAC Network. In: Proceedings of The 18th International Conference on Pattern Recognition, vol. 4, pp. 639–642 (2006)
9. John, D.H., Fischer, J., Johnam, D.: A Neural Network Pole Balancer that Learns and Operates on a Real Robot in Real Time. In: Proceedings of the MLC-COLT Workshop on Robot Learning, pp. 73–80 (1994)
10. Tatikonda, R.C., Battula, V.P., Kumar, V.: Control of Inverted Pendulum using Adaptive Neuro Fuzzy Inference Structure. In: IEEE Internal Symposium on Circuits and Systems, pp. 1348–1351 (2010)
11. Tetsuya, M., Furukawa, T.: The Self-Organizing Adaptive Controller. International Journal of Innovative Computing, Information and Control 7(4), 1933–1947 (2011)
12. Lin, C.J., Lee, C.Y.: Non-linear System Control using a Recurrent Fuzzy Neural Network based on Improved Particle Swarm Optimisation. International Journal of Systems Science 41(4), 381–395 (2010)
13. Hercus, R.G.: Neural networks with learning and expression capability. U. S. Patent 7412426 B2 (2008)
14. NeuraBase Generic Toolbox, `http://neuramatix.com/`
15. Cazzolato, B.S., Prime, Z.: On the Dynamics of the Furuta Pendulum. Journal of Control Science and Engineering (2011)

Coordinated Rule Acquisition of Decision Making on Supply Chain by Exploitation-Oriented Reinforcement Learning
-Beer Game as an Example-

Fumiaki Saitoh[1] and Akihide Utani[2]

[1] Department of Industrial and Systems Engineering,
College of Science and Engineering, Aoyama Gakuin University
5-10-1 Fuchinobe, Chuo-ku, Sagamihara City, Kanagawa, Japan
`saitoh@ise.aoyama.ac.jp`

[2] Department of Information and Communication Engineering,
Faculty of Knowledge Engineering, Tokyo City University
1-28-1, Tamadutumi, Setagaya-ku, Tokyo, Japan

Abstract. Product order decision-making is an important feature of inventory control in supply chains. The beer game represents a typical task in this process. Recent approaches that have applied the agent model to the beer game have shown. Q-learning performing better than genetic algorithm (GA). However, flexibly adapting to dynamic environment is difficult for these approaches because their learning algorithm assume a static environment. As exploitation-oriented reinforcement learning algorithm are robust in dynamic environments, this study, approaches the beer game using profit sharing, a typical exploitation-oriented agent learning algorithm, and verifies its result's validity by comparing
performances.

Keywords: reinforcement learning, agent based modeling, supply chain manegiment(SCM), beer game, exploitation-oriented learning.

1 Introduction

As markets and production continue to globalize, the importance of cooperative decision-making among companies within supply chains increases. Adjusing inventory order volumes from an upper process to a lower process often emerges as a decision-making problem. In inventory control, the costs of manageing over stock and the saless-opportunities lost through stock shortages generally have a trade-off relationship. A supply chain is a complex system composed of many companies that interact during decision-making. Thus, a mixture of partially optimal solutions provided by supply chain's components does not necessarily result in an optimal global situation.

Many recents studies have shown that computational intelligence methodologies, such as agent learning offer effective appraches supply chain problems[1][2].

© Springer-Verlag Berlin Heidelberg 2013

Reinforcement learning[3] is a particularly useful tool for global optimization of supply chain models through agent behavior. The research[6][5] suggestes that Q-learning[4] is superior to genetic algorithm (GA)[7],in assisting the optimization of supply chain task agents.

Reinforcement learning algorithm are generally classified into exploitation-oriented learning (such as Profit Sharing) and exploration-oriented learning (such as Q-learning). An advantage of exploration-oriented learning is that the optimal action rules can be acquired, through large amount of trial and error performed in a static environment. A great deal of trial and error is necessary, however, because these learning algorithm acquire rules through the identification of the environments; furthermore, their robustness is low in dynamic environments. On the other hand, exploitation-oriented learning algorism, which do not sacrifice optimality, have two important advantages: they require little trial and error to obtain semi-optimal action rules, and they are robust in dynamic environments.

Obviously, every company's environment is dynamic. A supply chain system not only features complex interactions among its components, but is also affected by what is exterior to it. Despite this fact, previous supply chain management studies have focused on Q-learning, an exploration-oriented learning weak in dynamic environments.

This study proposes the methodology of cooperative policy acquisition through exploitation-oriented reinforcement learning by employing the Beer Game, a well known supply chain role-playing game,in an experiment. We also confirm the effectiveness of our proposal through comparative experiments on exploitation-oriented reinforcement learning and exploration-oriented reinforcement learning using the game.

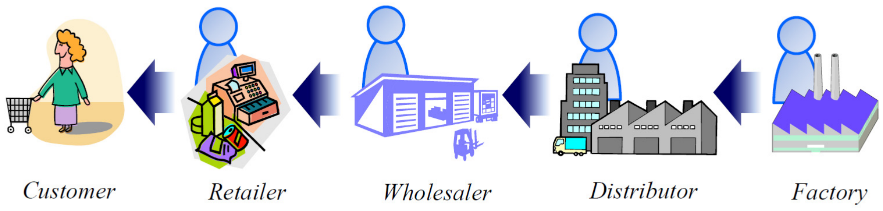


Fig. 1. A simple schematic diagram of a supply chain

2 Beer Game

2.1 Outline of the Beer Game

The Beer Game is a simple role-playing game designed to model the complexity of logistical dynamics. Its difficulty is in its prohibition of information-sharing among the players. When players who cannot share information make decisions in a given sequence, information on market demand deteriorates as the game progresses. Furthermore, because each player's decision-making results involve the other player's decision-making, the dynamics of the whole supply chain system becomes very complex. A Beer Game session is played by four players. Each player chooses a from among "retailer," "wholesaler," "distributor," "factory." A schematic diagram of the game, showing the flow from the upper process of the produced beer crates to the lower process, is shown in Fig.2.

Customer demand is determined when the retailer draws the card on which the order volume is written. The player who receives the order with in the lower process gives it to the player a single step above in the upper process. For example, after noting the order quantity given by retailer, the wholesaler places an order with the distributer, and the distributer then places an order with the factory. Throughout this process, communication between each player is inhibited, and key information, such as inventory quantity is unknown. Therefore, players must raise the profits of the whole supply chain, in spite of limited information.

Each player's penalties are imposed on the volume of inventories and the order backlog. The aim of the game is to minimize the total-cost of the supply chain. In order to enhance the throughput of the whole system, players must make decisions that do not generate an order backlog and allow them to hold a minimum inventorys. A player decision-making step, such as an order or delivery, is defined as one week. A game is terminated after all players have taken their turns for 50 weeks.

2.2 The Cardinal Rule of the Beer Game

The purpose of the game is to make decisions that minimize the penalty imposed on the whole supply chain. Since superfluous stock may weigh on management, a cost of 0.5$/week is imposed per unit of stock as the "ginventory cost." Moreover, since a stock deficit leads to lost sales opportunities, a cost of 1.0$/week is imposed per missing item as the "gshortage cost." The total cost accrued by all the players is the efficiency level of the whole supply chain.

Only the retailer knows the amount demanded of the customers. Other players know only the volume of the orders coming up from a single step below them. Therefore, the time lag by the time an order reaches a player is two weeks (see Fig.2). In Fig.2, the pink dots in the large blue box are beer cases owned by each player as stock. The pink dots in the small blue box are the shipped beer cases before they have reach the next player; these are treated as product shipment delays.

2.3 Procedure

The Beer Game proceeds as follows. In a week (a step), each player (agent) carrieds out an order and shipment in the way described below:.

i) The product in a shipment delay is carried to the next box.
ii) A player turns over an order card, and the number of products written on this card is shipped from the stock.

If an order backlog remains from the previous week, the player must ship both the number written on the order card and backlog from last week. If number of stocks is insufficient, only the existing stock will be shipped, and the insufficiency will be noted as an order backlog

iii) The amount of inventory or order backlog at this time is recorded.
iv) The card of the number required is moved to the next square.
v) A player determines the number required based on the amount of the current inventory, order backlog, or other amount.

Each player writes the number required on a card, turns it over, and passes it to the following square as an order. Each player then records the number required.

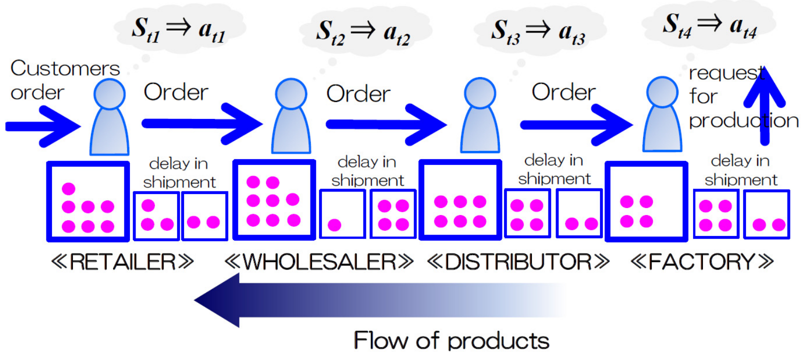


Fig. 2. Schematic diagram of the Beer Game

3 Exploitation-Oriented Reinforcement Learning

Exploitation-oriented reinforcement learning attaches importance not to the identification of an environment but to the efficiency of reward acquisition. We will not describe Profit Sharing, one of most common algorithm of exploitation-oriented reinforcement learning. State-action pairs used for each episode are memorized and employed to update a rule series. The reinforcement value for updating the rule set is calculated through the reinforcement function in Eq.(1). The reinforcement value used to update the rules is attenuated by Eq.(1) based on the oldness of the rule. The reinforcement function is usually expressed by the equal ratio monotone decreasing function as follow shown below:

$$f_n = \frac{1}{S} f_{n-1}, (n = 1, 2, ...W - 1) \tag{1}$$

where, W is the total number of steps in each episode, f_i is the reward value to the i-th rules, $1/S$ is the decreasing ratio, and S fulfills $S \geq L + 1$ when the number of effective rules in the identical state is L. The value obtained from subtracting 1 from the number of actions is generally sufficient as the value of L.

When a reward is acquired, the rule's weights are updated in Eq.(2), as below:

$$\omega(s_i, a_i) \leftarrow \omega(s_i, a_i) + r \times f_i \tag{2}$$

were, $\omega(s_i, a_i)$ is the weight of the i-th rule in a rule series, s_i is the state in the i-th step, and a_i is the state in the i-th step.

As all rules are updated independently in Profit Sharing, it is easy to converge the weight of the rules used frequently and contributeing to reward acquisition. Therefore, in addition to having a fast learning speed, Profit Sharing is robust in dynamic environments. Actions are selected based on the weight of the rule set updated by Eq.(2). A probabilistic policy, such as Boltzmann or roulette wheel selection is generally used.

4 Beer Game Optimization Using Exploitation-Oriented Reinforcement Learning

We will now apply Profit Sharing a typical form of exploitation-oriented reinforcement learning, to a decision-making problem in the Beer Game. The set of rules used in this task is constituted by state and action pairs. The objective function, reward, and state sets and action constituting if-then rules were defined as described below in order to formulate a task.

Action: The order quantity(Act_q) is selected from six options calculated by Eq.(3):

$$Act_q = X \times Ordered + Y \times 4 \tag{3}$$

where $X = (0, 1, 2)$,$Y = (0, 1)$,and *Ordered* is order quantity.

States: the number of stocks and order backlogs for each player

Objective Function: As the purpose of the game is to minimizeing the supply chain's total cost, this task is formulated as follows:

$$Minimize : Cost = \sum_{t=1}^{n} \sum_{i=1}^{4} [\alpha h_i(t) + \beta C_i(t)] \tag{4}$$

where $h_i(t)$ is quantity of stock and $C_i(t)$ is quantity of order backlog.

$$h_i(t) = \begin{cases} S_i(t) \; if(S_i > 0) \\ 0 \quad otherwise \end{cases} \tag{5}$$

$$C_i(t) = \begin{cases} |S_i(t)| \; if(S_i < 0) \\ 0 \quad otherwise \end{cases} \tag{6}$$

Reward: Here, we define the reward function as Eq.(7). It is important to keep total cost down and increase the number of products shipped during each episode. Therefore, reward is defined using total cost as follows:

$$r = \begin{cases} Rew \;\; if(cost < \mu) \\ -Rew \; if(cost > \nu) \\ \frac{\tau p}{cost} \;\; otherwise \end{cases} \tag{7}$$

where r is reward in each episode, p the number of products shipped in that episode, τ the parameter of the reward function, μ the desired value of cost, and ν the upper limit value of cost. Reward value Rew is given to agents when the cost reaches a desired value. When the cost exceeds nu, agents will be given a negative reward as a penalty.

5 Experiment

The validity of the proposed method was confirmed through the computational simulation, details of which are given below.

5.1 Experimental Settings

In the experiment, Beer Game players were transposed into agents, and the performances of the two learning algorithm. Profit Sharing and Q-learning, were compared. Agents traded once a week, which we defined as one step. The period from the first to the fiftieth week was defined as one episode, We calculated 100,000 episode iterations on the computer. The volume of inventories and order backlogs for each player were used to express the state in the "*if-then*" rule showing the action. Roulette wheel selection was used to select the actions. In each episode, each agent had an initial inventory volme of 12, and the initial inventory volume in the shipment delay square wasfour.

Previous studise have used static data with little or no value changes to represent the customer demand number. As Poisson distribution is a well-known demand distribution, we used a Poisson random number as the demand quantity. The parameters used in the experiment are shown in Table1.

5.2 Experimental Results

The results of the experiments are summarized in Fig.3. The horizontal line represents the number of episodes, and the vertical line, the number of total costs per episode. To highligt the difference between the two algorithm, Fig.3 presents a double-logarithmic chart. As the number of episodes increases, the degree of the fluctuation seems to grow but not in the double-logarithmic chart. The task performances for each method can be evaluated by comparing their total cost transition. These lines express the average value of the results of the thirtieth iteration To make the experimental result legible, all the lines are smoothed by the moving average, the section width of which is 101 steps. In Fig.3, the learning curve of Q-learning appears as the blue line, and that of Profit Sharing, as the pink line.

Table 1. List of parameters

Q-leaerning	
Learning rateα	0.1
Discount factorγ	0.9
Number of states	21
Profit Sharing	
Discount rate of reward$1/S$	0.5
Number of states	21
Poisson distribution	
λ	3
Parameter of reward	
μ	6000
ν	14000
Rew	1
τ	10

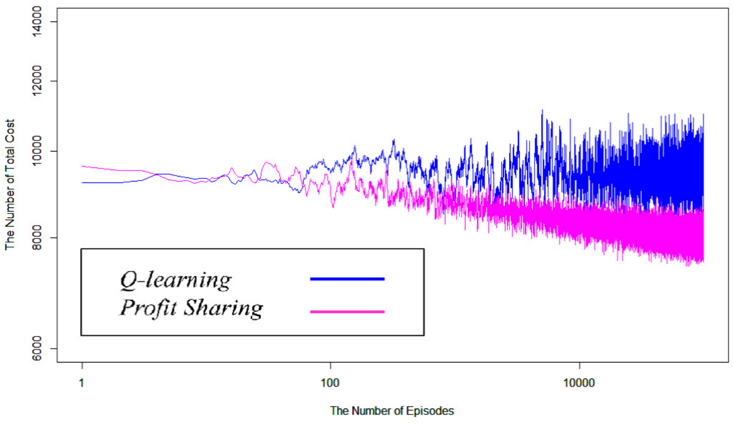


Fig. 3. Experimental result. The pink line is the learning curve of Profit Sharing, and the blue line is the learning curve of Q-learning.

5.3 Discussion

First, comparing the learning curve of Q-learning with that of Profit Sharing shows that the converging profit sharing on the value of the solution is better in the latter.

Next, examining the result of Q-learning (the blue line) reveals that the Q-learning solution did not converge easily; the number of states is insufficient, and the learning algorithm is not suitable for a dynamic environment.

Finally, we consider the result of Profit Sharing (the pink line). As the cost amount begins to decrease in the experiment's early stages, the earliness of Profit Sharing's start can be confirmed. Despite the lack of other agents' information, Profit Sharing's learning result in this dynamic Beer Game task converges on a semi-optimal solution. These results indicate that exploitation-oriented learning is superior to exploration-oriented learning in supply chain optimization.

We plan to perform a more detailed experiment an investigation of the parameter's influence on the experimental result. We could also improve performance by increasing the information used for the state variable. We believe that information's state space construction affects the performance of reinforcement learning in supply chain management.

6 Conclusions

This study, proposed that an agent-based decision making model for the Beer Gamebe applied to the dynamics of the supply chain. The action acquisition of exploration-oriented learning is not suitable for the Beer Game because of its lack of robustness and slow learning speed. As an alternative, we have proposed a methodology using the exploitation-oriented learning algorithm. The effectiveness of our proposal was experimentally confirmed through a comparison between Profit Sharing and Q-learning. The simulation result shows that exploitation-oriented learning is superior to exploration-oriented learning in a dynamic environment like a supply chain. Future research will address the development of this algorithm and apply it to practical cases such as the tasks involved in large and complicated supply chains.

References

1. Stockheim, T., Schwind, M., Koenig, W.: A Reinforcement Learning Approach for Supply Chain Management. In: 1st Europian Workshop on Malti Agent Systems (2003)
2. Kimbrough, S.O., Wu, D.J., Zhong, F.: Computer play the beer game-can artificial agents manage supply chains? Decision Support Systems 33(3), 323–333 (2002)
3. Sutton, R., Barto, A.: Reinforcement Learning. MIT Press (1998)
4. Watkins, C.J.H., Dayan, P.: Technical note: Q-learning. Machine Learning 8, 55–68 (1992)
5. van Tongeren, T., Kaymak, U., Naso, D., van Asperen, E.: Q-Learning in a Competitive Supply Chain. In: IEEE International Conference on Systems, Man and Cybernetics, ISIC, pp. 1211–1216 (2007)
6. Kamal Chaharsooghi, S., Heydari, J., Hessameddin Zegordi, S.: A reinforcement learning model for supply chain ordering management-An application to the beer game. Decision Support Systems 45(4), 949–959 (2008)
7. Grefenstette, J.J.: Credit Assignment in Rule Discovery System Based on Genetic Algorithms. Machine Learning 3, 225–245 (1988)
8. Arai, S., Miyazaki, K., Kobayashi, S.: Methodology in Multi-Agent Reinforcement Learning: Approaches by Q-Learning and Profit Sharing. Transaction of the Japanese Society for Artificial Intelligence 13(4), 609–618 (1998) (in Japanese)
9. Iyer, A., Seshadri, S., Vasher, R.: Toyotafs Supply Chain Management: A Strategic Approach to Toyota's Renowned System. McGraw-Hill Education (2009)
10. Ichikawa, M., Koyama, Y., Deguchi, H.: Human and Agent Playing the"Beer Game". Developments in Business Simulation and Experiential Learning 35, 231–237 (2008)

Using Exponential Kernel for Word Sense Disambiguation

Tinghua Wang[1,2], Junyang Rao[1], and Dongyan Zhao[1]

[1] Institute of Computer Science and Technology, Peking University, Beijing 100871, China
{wangtinghua,raojunyang,zhaodongyan}@pku.edu.cn
[2] School of Mathematics and Computer Science, Gannan Normal University, Ganzhou 341000, China

Abstract. The success of machine learning approaches to word sense disambiguation (WSD) is largely dependent on the representation of the context in which an ambiguous word occurs. Typically, the contexts are represented as the vector space using "Bag of Words (BoW)" technique. Despite its ease of use, BoW representation suffers from well-known limitations, mostly due to its inability to exploit semantic similarity between terms. In this paper, we apply the exponential kernel, which models semantic similarity by means of a diffusion process on a graph defined by lexicon and co-occurrence information, to smooth the BoW representation for WSD. Exponential kernel virtually exploits higher order co-occurrences to infer semantic similarities in an elegant way. The superiority of the proposed method is demonstrated experimentally with several SensEval disambiguation tasks.

Keywords: Word sense disambiguation (WSD), Exponential kernel, Support vector machine (SVM), Kernel method, Natural language processing.

1 Introduction

Word sense disambiguation (WSD) refers to the task of identifying the correct sense of an ambiguous word in a given context [1]. As a fundamental semantic understanding task at the lexical level in natural language processing, WSD can benefit many applications such as information retrieval and machine translation. However, it has been very difficult to formalize the process of disambiguation, which humans can do so effortlessly. Generally, there are two main kinds of methods to perform the task of WSD: knowledge-based approaches and corpus-based approaches. The former disambiguate words by comparing their context against information from the predefined lexical resources such as WordNet, whereas the latter do not make use of any these resources for disambiguation [1]. Most of the corpus-based approaches stem from the machine learning community, ranging from supervised learning in which a classifier is trained for each distinct word on a corpus of manually sense-annotated examples, to completely unsupervised

© Springer-Verlag Berlin Heidelberg 2013

methods that cluster occurrence of words, thereby inducing senses. Moreover, in recent years it seems very promising that applying kernel methods [2] such as Support Vector Machine (SVM) [3–6] and Regularized Least-Squares Classifier (RLSC) [7] to the WSD task. The advantage of using kernel methods is that they offer a flexible and efficient way of defining application-specific kernels for introducing background knowledge and modeling explicitly linguistic insights.

For the machine learning-based WSD, one of the key steps is the representation of the context of the target ambiguous word. In the commonly used ”Bag of Words (BoW)” representation [2], contexts are represented by vectors whose dimensions are indexed by different words or terms occurring in the contexts. Despite its ease of use, the BoW representation suffers from well-known limitations, mostly due to its inability to exploit semantic similarity between terms: contexts sharing terms that are different but semantically related will be considered as unrelated. The lack of semantics in the BoW representation limits the effectiveness of automatic WSD. To alleviate this shortage, a number of attempts have been made to incorporate semantic knowledge into the BoW representation. For example, the external semantic knowledge provided by word thesauri or ontology was embedded into a semantic kernel, which is used to enrich the standard BoW paradigm [5]. In the absence of external semantic knowledge, corpus-based statistical methods, such as Latent Semantic Indexing (LSI) [8] can be applied to capture semantic relations between terms [6,9]. However, such methods are also limited in their flexibility and usually computationally expensive.

In this paper, we present and evaluate a semantically-enriched BoW representation for WSD. We adopt the exponential kernel [10] to efficiently model semantic similarity by means of a diffusion process on a graph defined by lexicon and co-occurrence information. The idea behind this method is that higher order correlations between terms can affect the semantic similarities. Combined with the use of SVM, this kernel shows a significant disambiguation improvement over the standard BoW kernel. To the best of our knowledge, our work is the first time to apply this kernel to the WSD application.

2 Kernels Based on BoW Representation

WSD can be viewed as a problem of classifying each word, according to its surrounding context, to one of its senses. Therefore it is a classification problem with a few classes. A key step in applying SVM to WSD is to choose an appropriate kernel function. Up to now, most methods tried to represent the contextual information in a vector, and then used some standard vector-based kernels. In natural language processing, it is widely agreed that the linear kernel performs better than other kernels [5,7]. In this section, we will discuss such vector-based data representation and kernels used for WSD.

Let t denote a word to be disambiguated and $\boldsymbol{d} = (t_{-r}, \cdots, t_{-1}, t_1, \cdots, t_s)$ be the context of t, where $t_{-r}, \cdots, t_{-1}$ are the words in the order they appear preceding t, and correspondingly $t_1, \cdots, t_s$ are the words that follow t in the text. We also define a context span parameter τ to control the length of the context. For a fixed τ, we take always the largest context $\boldsymbol{d}$ so that $r \leq \tau$ and $s \leq \tau$. Note that if there exist τ words preceding and following the word to be disambiguated, then $r = s = \tau$, otherwise $r < \tau$ or $s < \tau$. In addition, let n be the size of the set of all distinct words of all the contexts in the training corpus. The BoW model [2] of a context $\boldsymbol{d}$ is defined as follows:

$$\phi : \boldsymbol{d} \to \phi(\boldsymbol{d}) = (tf(t_1, \boldsymbol{d}), \cdots, tf(t_n, \boldsymbol{d}))^{\mathrm{T}} \in R^n \tag{1}$$

where $tf(t_i, \boldsymbol{d})$, $1 \leq i \leq n$, is the frequency of the occurrence of word t_i in the context $\boldsymbol{d}$. The BoW kernel is given by:

$$k(\boldsymbol{d}_i, \boldsymbol{d}_j) =< \phi(\boldsymbol{d}_i), \phi(\boldsymbol{d}_j) >= \phi(\boldsymbol{d}_i)^{\mathrm{T}} \phi(\boldsymbol{d}_j) \tag{2}$$

In the BoW representation, the feature vectors are typically sparse with a small number of non-zero entries for those words occurring in the contexts. Two contexts that use semantically related but distinct words will therefore show no similarity. Ideally, semantically similar contexts should be mapped to nearby positions in the feature space. In order to address the semantic information of the words in BoW, a transformation of the feature vector of the type $\bar{\phi}(\boldsymbol{d}) = \boldsymbol{S}\phi(\boldsymbol{d})$ is required, where $\boldsymbol{S}$ is a semantic matrix and indexed by pairs of terms with the entry $S_{i,j} = S_{j,i}$, $1 \leq i, j \leq n$, indicating the strength of their semantic similarity. Using this transformation, the semantic kernels take the form

$$k(\boldsymbol{d}_i, \boldsymbol{d}_j) = \bar{\phi}(\boldsymbol{d}_i)^{\mathrm{T}} \bar{\phi}(\boldsymbol{d}_j) = \phi(\boldsymbol{d}_i)^{\mathrm{T}} \boldsymbol{S}^{\mathrm{T}} \boldsymbol{S} \phi(d_j) \tag{3}$$

The semantic kernels correspond to representing a context as a less sparse vector, $\boldsymbol{S}\phi(\boldsymbol{d})$, which has non-zero entries for all terms that are semantically similar to those presented in context $\boldsymbol{d}$.

3 Exponential Kernel Applied to WSD

As mentioned above, the key problem of a supervised WSD system based on SVM is how to choose the appropriate kernel. In the framework of semantic kernels, this problem reduces to how to choose the semantic matrix $\boldsymbol{S}$. In this section we will discuss a sophisticated method to solve this problem.

3.1 Exponential Kernel

The problem of how to infer semantic relations between terms from a corpus remains an open issue. Kandola et al. [10] modeled semantic relations by means of a diffusion process on a graph defined by lexicon and co-occurrence information and derived a semantic diffusion kernel named exponential kernel given by:

$$\tilde{\boldsymbol{K}}(\lambda) = \boldsymbol{K} \exp(\lambda \boldsymbol{K}) \tag{4}$$

where $\boldsymbol{K}$ is the kernel matrix (Gram matrix) of the BoW kernel and λ is a decay factor. Let $\boldsymbol{D}$ be feature example (term-by-context in the case of WSD) matrix in the BoW kernel-induced feature space, then $\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}$ gives the kernel matrix $\boldsymbol{K}$.

Let $\boldsymbol{G} = \boldsymbol{D}\boldsymbol{D}^{\mathrm{T}}$, it has been proved that $\tilde{K}(\lambda)$ corresponds to a semantic matrix $\exp(\lambda\boldsymbol{G}/2)$ [2,10], i.e.,

$$\boldsymbol{S} = \exp\left(\frac{\lambda}{2}\boldsymbol{G}\right) = \frac{1}{2}\left(2\boldsymbol{I} + \lambda\boldsymbol{G} + \frac{\lambda^2\boldsymbol{G}^2}{2!} + \cdots + \frac{\lambda^\theta\boldsymbol{G}^\theta}{\theta!} + \cdots\right) \tag{5}$$

In fact, noting that $\boldsymbol{S}$ is a symmetric matrix since $\boldsymbol{G}$ is symmetrical, we have

$$\begin{aligned}\tilde{\boldsymbol{K}}(\lambda) &= \boldsymbol{D}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{D} = \boldsymbol{D}^{\mathrm{T}}\boldsymbol{S}^2\boldsymbol{D} = \boldsymbol{D}^{\mathrm{T}}\exp(\lambda\boldsymbol{G})\boldsymbol{D}\\ &= \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} + \lambda\boldsymbol{D}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{D} + \frac{\lambda^2\boldsymbol{D}^{\mathrm{T}}\boldsymbol{G}^2\boldsymbol{D}}{2!} + \cdots + \frac{\lambda^\theta\boldsymbol{D}^{\mathrm{T}}\boldsymbol{G}^\theta\boldsymbol{D}}{\theta!} + \cdots\\ &= \boldsymbol{K}\left(\boldsymbol{I} + \lambda\boldsymbol{K} + \frac{\lambda^2\boldsymbol{K}^2}{2!} + \cdots + \frac{\lambda^\theta\boldsymbol{K}^\theta}{\theta!} + \cdots\right)\\ &= \boldsymbol{K}\exp(\lambda\boldsymbol{K})\end{aligned} \tag{6}$$

where $\boldsymbol{I}$ denotes the identity matrix.

The semantic matrix $\boldsymbol{S}$ essentially captures the higher order correlation between terms. Conceptually, if term t_1 co-occurs with term t_2 in some contexts, we say t_1 and t_2 share a first-order correlation between them. If t_1 co-occurs with t_2 in some contexts, and t_2 with t_3 in some others, then t_1 and t_3 are said to share a second-order correlation through t_2. Higher orders of correlation may be similarly defined. Consider Figure 1(a) depicting three contexts $\boldsymbol{d}_1$, $\boldsymbol{d}_2$ and $\boldsymbol{d}_3$, each containing two terms from t_1, t_2, t_3 and t_4. We can find that t_1 and t_4 share a third-order correlation through t_2 and t_3. When modeled as a graph as shown in Figure 1(b), each higher order correlation defines a path between the two vertices (terms). For the semantic matrix $\boldsymbol{S}$, the entries in the matrix $\boldsymbol{G}^\theta$ are given by

$$G^\theta_{i,j} = \sum_{\substack{m_1,\cdots,m_\theta \in \{1,\cdots,n\}\\ m_1=i, m_\theta=j}} \prod_{p=1}^{\theta-1} G_{m_p,m_{p+1}} \tag{7}$$

that is the number of θth-order co-occurrence paths between terms i and j. Hence the semantic similarity between two terms is measured by the number of the co-occurrence paths between them. Specifically, $\boldsymbol{G}$ indicates the first-order correlation between features (terms) over the training corpus, $\boldsymbol{G}^2$ indicates the second-order correlation between terms, and so forth, $\boldsymbol{G}^\theta$ indicates the θth-order correlation between terms. In addition, it should be noted that the identity matrix $\boldsymbol{I}$, which can be regarded as the indication of the zero-order correlation between terms, means only the similarity between a term and itself equals 1 and 0 for other cases. Intuition shows that the higher the co-occurrence order is, the less similar the semantics becomes. The semantic matrix $\boldsymbol{S}$ combines all the order co-occurrence paths with exponentially decaying weights and the parameter $\lambda \in [0, +\infty)$ is used to control the decaying speed for increasing orders. Finally, it is easy to find that the exponential kernel is reduced to the standard BoW kernel when $\lambda = 0$. In other words, the exponential kernel is just a generalization of the BoW kernel.

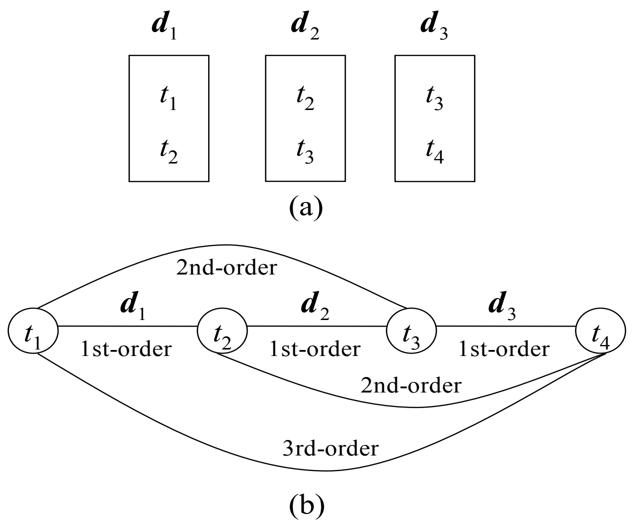


Fig. 1. Graphical representation of higher order co-occurrences

3.2 Discussion

Many current natural language processing systems rely on semantic knowledge extracted from the data via machine learning methods. The main problem encountered by such systems is the *sparse data problem*, due to the small amount of learning examples. Focusing on WSD, only a handful of occurrences with sense tags are available per word. One possible solution to this problem is to utilize higher order co-occurrences as a measure of semantic relation between terms. The underlying idea is that humans do not necessarily use the same vocabulary when writing about the same topic [11].

Several approaches in the past few years have been proposed to model term similarity based on the concept of higher-order co-occurrence [12,13]. Perhaps the most sophisticated and widely used method is LSI, which was originally applied to information retrieval. Given a term-by-context matrix $\boldsymbol{D}$, the general idea of LSI is to reduce $\boldsymbol{D}$ using Singular Value Decomposition (SVD) technique which is thought to reduce the noise in the data and make relationships between underlying concepts more clear. In fact, LSI takes advantage of implicit higher order (or latent) structure in the association of terms and contexts [13]. LSI-based kernels, such as latent semantic kernels [9] and domain kernels [6], have also been investigated. Conceptually, the domain kernel estimates the similarity between the domains of two texts, so to capture domain aspects of sense distinction. It is a variation of the latent semantic kernel.

The main problem of LSI is its high computation complexity since computing the SVD itself is not trivial. For a $n \times n$ matrix, the SVD computation requires time proportional to n^3. This is impractical for matrices with more than a few thousand dimensions. In comparison with LSI, exponential kernel does not need the heavy computational overhead of decomposition. In addition, exponential kernel has the advantage of explicitly capturing the latent or higher order similarities, as opposed to doing that implicitly as in LSI. Finally, it should be noted that, more recently, there is increasing evidence that Independent Component Analysis (ICA) is more efficient and accurate than LSI/SVD and used as the first step in the text analysis [14]. ICA applied on word context data can give distinct features which reflect linguistic categories.

4 Evaluation Results

This section provides an empirical study to show the benefits of the proposed WSD system with several real corpora. Specifically, we compare the performance for three methods: 1) **MFS**: a common baseline model that selects the most frequent sense (MFS) in the training data as the answer. 2) **SVM_BoW**: SVM classifier with BoW kernel. 3) **SVM_EK**: SVM classifier with exponential kernel. We use the SVM since it has been shown to achieve the best results compared to several supervised approaches [3].

We select the corpora for four words, namely *interest*, *line*, *hard* and *serve*, which have been used in numerous comparative studies of WSD, from the SensEval website[1]. There are 2368 instances of *interest* with 6 senses, 4147 instances of *line* with 6 senses, 4333 instances of *hard* with 3 senses, and 4333 instances of *serve* with 4 senses. For each corpus, we partition it into a training set and a test set by stratified sampling: 70% of the data set serves as training set and 30% as test set. For the training set, we first remove the words that are in a list of stop words (for example: "is", "a", "the"). Words that contain no alphabetic characters, such as punctuation symbols and numbers, are also discarded. We then extract the surrounding words, which can be in the current sentence or immediately adjacent sentences, in the ±5-word window (i.e., $r = s = \tau = 5$) context of an ambiguous word. The extracted words are finally converted to their lemma forms in lower case. Each lemma is considered as one feature and whose value is set to be the "term frequency (TF)". For the test set, the similar preprocessing is carried out but the features are the same as those extracted from the training set (we directly eliminate those lemmas found in the test set but not in the training set).

Once the training and test instances are represented as feature vectors, we trains an SVM classifier for each word type with the training data and tests the disambiguation performance of the learned model with the test data. The parameters of the SVM are optimized by five-fold cross-validation on the training set. When using the BoW kernel, there is only one parameter C (the regularization parameter used to impose a trade-off between the training error and generalization in SVM) need to be optimized. We perform grid-search in one dimension (i.e., a line-search) to choose this parameter from the set $\{2^{-2}, 2^0, \cdots, 2^{10}\}$. When using the exponential kernel, there are two parameters C and λ need to be optimized. We perform grid-search over two dimensions, i.e., $C = \{2^{-2}, 2^0, \cdots, 2^{10}\}$

[1] `http://www.senseval.org/`

and $\lambda = \{2^0, 2^{-1}, \cdots, 2^{-10}\}$. In addition, the implementation of the SVM classifier is achieved by the software LIBLINEAR[2] [15], which is a simple but efficient package for solving large-scale linear classification and regression problems.

Table 1. Disambiguation performances of three methods

Corpus	Classification accuracy (%)		
	MFS	SVM_BoW	SVM_EK
interest	52.8716	85.9598	87.3232
line	53.4844	82.9445	84.0881
hard	79.7369	83.6972	84.8374
serve	41.4344	86.2909	86.9956

We report the results in terms of the standard measure of *classification accuracy*, which indicates the percentage of instances of an ambiguous word that were correctly classified from the entire test set. The average accuracies over 10 trials are summarized in Tab. 1. We can see that both SVM_BoW and SVM_EK achieve significantly better accuracies than the MFS baseline. More importantly, SVM_EK successfully outperforms SVM_BoW: the accuracies increase from 85.9598% to 87.3232%, 82.9445% to 84.0881%, 83.6972% to 84.8374% and 86.2909% to 86.9956% for disambiguating the word *interest*, *line*, *hard* and *serve*, respectively. This demonstrates the effectiveness of the proposed SVM_EK method. It should be noted that the performance differences are statistically significant ($p>0.05$) in light of the pairs t-tests on all four corpora.

5 Conclusion

We have explored the use of exponential kernel for improving the performance of SVM classifier in word sense disambiguation. Geometrically, exponential kernel models semantic similarities as a diffusion process in a graph whose nodes are the terms and edges incorporate the first-order similarity. Diffusion efficiently takes all the possible paths connecting two nodes into account, and propagates the similarity between two remote terms. Empirical evaluation on several SensEval tasks demonstrates that our approach successfully improves the disambiguation performance compared to that using the standard BoW kernel. Moreover, in a supervised WSD framework, we have class information of training data in addition to the co-occurrence information. An inherent limitation of the exponential kernel is that it fails to exploit the class information of training data. It will be an interesting future work to present new approaches or apply existing technologies such as “sprinkling” [12] to refine the exponential kernel.

Acknowledgments. This work is supported in part by the National High Technology Research & Development Program of China (No. 2012AA011101), the National Natural Science Foundation of China (No. 61202265), the China Postdoctoral Science Foundation Funded Project (No. 2012M510275) and the Natural Science Foundation of Jiangxi Province of China (No. 20114BAB211021).

[2] `http://www.csie.ntu.edu.tw/~{}cjlin/liblinear`

References

1. Navigli, R.: Word Sense Disambiguation: A Survey. ACM Computing Surveys 41(2), 1–69 (2009)
2. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York (2004)
3. Lee, Y.K., Ng, H.T.: An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Philadelphia, USA, pp. 41–48 (2002)
4. Lee, Y.K., Ng, H.T., Chia, T.K.: Supervised Word Sense Disambiguation with Support Vector Machines and Multiple Knowledge Sources. In: Proceedings of Senseval–3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, pp. 137–140 (2004)
5. Jin, P., Li, F., Zhu, D., Wu, Y., Yu, S.: Exploiting External Knowledge Sources to Improve Kernel-based Word Sense Disambiguation. In: Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering, Beijing, China, pp. 1–8 (2008)
6. Giuliano, C., Gliozzo, A., Strapparava, C.: Kernel Methods for Minimally Supervised WSD. Computational Linguistics 35(4), 513–528 (2009)
7. Popescu, M.: Regularized Least-squares Classification for Word Sense Disambiguation. In: Proceedings of Senseval–3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, pp. 209–212 (2004)
8. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
9. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent Semantic Kernels. Journal of Intelligent Information Systems 18(2-3), 127–152 (2002)
10. Kandola, J., Shawe-Taylor, J., Cristianini, N.: Learning Semantic Similarity. In: Advances in Neural Information Processing Systems, vol. 15, pp. 657–664 (2003)
11. Lemaire, B., Denhière, G.: Effects of High-order Co-occurrences on Word Semantic Similarity. Current Psychology Letters 18(1) (2006)
12. Chakraborti, S., Wiratunga, N., Lothian, R., Watt, S.: Acquiring Word Similarities with Higher Order Association Mining. In: Proceedings of the 7th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development, Belfast, UK, pp. 61–76 (2007)
13. Kontostathis, A., Pottenger, W.M.: A Framework for Understanding Latent Semantic Indexing (LSI) Performance. Information Processing and Management 42(1), 56–73 (2006)
14. Honkela, T., Hyvärinen, A., Väyrynen, J.: WordICA-Emergence of Linguistic Representations for Words by Independent Component Analysis. Natural Language Engineering 16(3), 277–308 (2010)
15. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research 9, 1871–1874 (2008)

Independent Component Analysis Filtration for Value at Risk Modelling

Ryszard Szupiluk[1], Piotr Wojewnik[1], and Tomasz Ząbkowski[2]

[1] Warsaw School of Economics, Al. Niepodleglosci 162, 02-554 Warsaw, Poland
[2] Warsaw University of Life Sciences, Nowoursynowska 159, 02-776 Warsaw, Poland
rszupi@sgh.waw.pl, piotr.wojewnik@gmail.com,
tomasz_zabkowski@sggw.pl

Abstract. In this article we present independent component analysis (ICA) applied to the concept of value at risk (VaR) modelling. The use of ICA decomposition enables to extract components with particular statistical properties that can be interpreted in economic terms. However, the characteristic of financial time series, in particular the nonstationarity in terms of higher order statistics, makes it difficult to apply ICA to VaR right away. This requires using adequate ICA algorithms or their modification taking into account the statistical characteristics of financial data.

Keywords: Value at Risk, Independent Component Analysis, financial time series analysis.

1 Introduction

Risk modelling is one of the fundamental problems in the investment theory and practice on financial markets [7]. One of the most popular methods is the concept of value at risk (VaR) [6,9]. VaR over a given time horizon can be defined in the following way:

$$P(W \leq W_0 - VaR) = \alpha, \tag{1}$$

where W is portfolio value at the end of analyzed period (random variable), W_0 is present portfolio value and α is risk level. For the return rates VaR can be determined as:

$$VaR = -R_\alpha W_0, \tag{2}$$

where R_α is the quantile of return rates distribution at given risk level (probability of risk). Although the concept itself is simple and intuitive, it is associated with a fundamental problem of estimating the probability that a given financial instrument will reach specific values in the future. As a result, VaR concept is quite closely related to forecasting the future values of a financial instrument. In this area, one of

© Springer-Verlag Berlin Heidelberg 2013

the most commonly used approaches are these based on simulations [10]. They aim to find the possibly best mathematical model for the instrument based on historical data, and then performing predictive simulation. As a result, the whole issue boils down to the choice of an adequate simulation model. This opens up a discussion on the adequacy of model fitting to the empirical data.

According to the above, in case of predictive modelling based on historical data, we can distinguish two dominant approaches. The first assumes that the data should not be modified, because they interfere with the representation of the real phenomena. The second approach allows for modification of underlying data to create predictive models and simulations, especially when we are interested in main trends without noisy fluctuations or unusual events. For instance, data modification can be motivated by the fact that rare events like 2008 crash will not happen very often (they appear occasionally by definition), and their presence in models may distort the reality.

In this research, using financial market data, we are primarily interested in the separation of the components which are typical for well-functioning and effective markets in contrast to the components related to noises or the specific rare events. As separation method we apply Independent Component Analysis which is one of the main methods used in Blind Source Separation (BSS) problem [3,8]. The typical application ICA in VaR concept is based on performing risk analysis directly on separated independent source components [2,18]. In our approach, we use ICA for filtration and elimination of specific signals that may highly over-influence the data analysis by some very untypical events.

However, this approach, based on the existing ICA, has some limitations related to high instability of higher order statistics observed in many financial time series. In this paper, we propose a way to solve this problem, along with a new version of Natural Gradient algorithm, which we call Local Natural Gradient.

2 ICA Decomposition for Separation and Filtration

Independent component analysis can be considered as one of blind signal separation methods, where the main aim is to find latent source signals hidden in their mixture [1,8]. The standard model for BSS as well ICA is described as

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)\,, \tag{3}$$

where $\mathbf{x} = [x_1,...,x_n]^T$ are observed signals, $\mathbf{s} = [s_1,...,s_m]^T$ are hidden (unknown) source signals, and $\mathbf{A} \in \Re^{n \times m}$ is unknown matrix representing the mixing system. The purpose of the ICA is proper separation (reconstruction, estimation) of the observed signals **x** into source signals **s,** accepting the ambiguity according to the scale and permutation of source signals. To find the solution we need such matrix **W** that for

$$\mathbf{y}(t) = \mathbf{W}\mathbf{x}\,(t) = \mathbf{W}\mathbf{A}\mathbf{x}\,\,(t) = \mathbf{P}\mathbf{D}\mathbf{s}\,\,(t)\,, \tag{4}$$

where **P** is permutation matrix to define the order of estimated signals and **D** is a diagonal scaling matrix [8].

The basic ICA method can be applied for filtration approach [17]. Under assumption that the source signal vector contains certain constructive components $\hat{s}_j$, for $j = 1,...,p$ and noises $\tilde{s}_l$, for $l = 1,...,q$ then we can write:

$$\mathbf{s}(t) = [s_1(t),..., s_n(t)]^T = [\hat{s}_1(t),..., \hat{s}_p(t), \tilde{s}_{p+1}(t)...., \tilde{s}_{p+q}(t)]^T \, . \tag{5}$$

After separation of latent components we reject the destructive ones (replacing them with zero $\tilde{s}_l = 0$) to obtain improved version $\hat{\mathbf{x}}$ of observed signals **x:**

$$\hat{\mathbf{x}}(t) = \mathbf{A}[\hat{s}_1(t),..., \hat{s}_p(t), 0_{p+1}(t)...., 0_n(t)]^T \, . \tag{6}$$

Depending on the data characteristics and the chosen separation method a certain set of assumptions should be adopted. In case of ICA, a fundamental premise assumes the independence of the source signals, which means that the separated signals should also be independent of each other. In practice, to extract independent components from observed signals some additional assumptions are required: (i) the columns of matrix **A** should be linearly independent; (ii) $n \geq m$ - number of observed signals cannot be lower than the number of independent source signals; (iii) the signals are modeled as random variables or stochastic white noises, where at most one signal s_i has Gaussian distribution.

There are couple of methods that can achieve (4). The common feature linking them is exploration of higher order statistics, in particular kurtosis. This is the case of JADE, FASTICA, and Natural Gradient algorithms [1,3,8]. However, to perform ICA decomposition, the optimal form of non-linearity in the ICA algorithms requires knowledge of the signals distribution, which is, in practice, usually impossible. This implies a certain set of surrogate actions, such as using models based on flexible distributions or selection of the nonlinear concepts exploring empirical characteristics of the data. One simple heuristic method for nonlinearity selection is based on the observation that this type of non-linearity takes the form of the linear function for the Gaussian distribution, which is growing faster for the super Gaussian (leptokurtic) distribution and growing slower for sub Gaussian distribution. An extension of this approach is parametric model for the separated signals, for which parameters and, consequently, non-linear functions are selected in adaptive learning process. The main problem of ICA methods applied to financial data is the fact that these time series are often characterized by unstable kurtosis, see Fig. 1.

These properties of financial time series significantly reduce the effectiveness of the ICA algorithms based on the typical nonlinearities. A possible solution to this problem is the use of models more thoroughly exploring kurtosis and skewness. One of the most general parametric models is the approach proposed by [12], which is based on Extended Generalized Lambda Distribution (EGLD) system to model the distributions with different kurtosis and skewness. EGLD system consists of two distributions: Generalized Lambda Distribution (GLD) and Generalized Beta Distribution (GBD) [11].

This non-linear model, although applicable for modelling signals with a wide range of kurtosis and skewness, does not meet completely the expectations to the non-stability problem. In case of non-stability and non-stationarity of the signals a reasonable approach could be on-line ICA algorithm [3]. However, in case of on-line ICA algorithms, we should pay attention to another specific aspect of financial time series which is kurtosis estimation based on the sample that is highly sensitive to outliers (unusual) values.

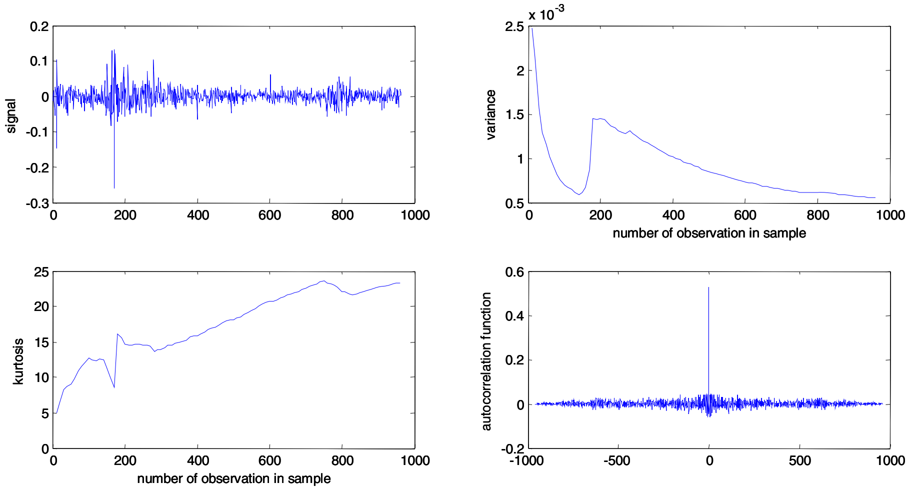


Fig. 1. Characteristics of Warsaw Stock Exchange Index (WIG) calculated on logarithmic return rates (including time interval from 04-01-2008 until 01-08-2012). Upper left figure is original signal, lower left figure is kurtosis, upper right figure is variance, and lower right is autocorrelation function.

However, while in case of typical technical signals (in engineering) outlier values are usually the result of measurement problems or noises, whereas in case of financial time series the abnormalities correspond to some important events appearing on the financial markets. Such events are apparent not only at that moment in time, but they influence the following values. As a result, we have some local dependencies, which are neither global nor individual. In such case, to perform effective ICA we propose following algorithm for matrix **W** finding

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \mu(t)\left[\mathbf{I} - \sum_{k=0}^{K} b_i f_i(\mathbf{y}(t))\mathbf{y}^T(t-k)\right]\mathbf{W}(t), \tag{7}$$

where

$$f_i(y_i) = -\frac{\partial \log(p_i(y_i))}{\partial y_i}. \tag{8}$$

The main idea of algorithm (7) is similar to spatio-temporal extensions of basic ICA algorithm, which is addressed for data with temporal structure [3,5,14,16].

The internal term in (7) with the form of:

$$\tilde{\mathbf{R}}_{yy} = \sum_{k=0}^{K} b_i f_i(\mathbf{y}(t))\mathbf{y}^T(t-k), \tag{9}$$

can be interpreted as weighted local non-linear covariance matrix. The motivation for this matrix application is similar to the approach in which time delay matrices combination are applied in second order statistics algorithms for blind source separation [4,13,15]. The main question is the choice of coefficients combination, what determine matrix (9) characteristics. The coefficients b_i choice may be based on the following variability (volatility) measure (*ksi*):

$$\xi(y) = \frac{\frac{1}{K}\sum_{k=0}^{K}\left|y(t-k) - y(t-k-1)\right|}{\max(y) - \min(y) + \delta(\max(y) - \min(y))}, \tag{10}$$

where symbol $\delta(.)$ means zero indicator function introduced to avoid dividing by zero (valued at 0 everywhere except 0, where the value of $\delta(.)$ is 1). Measure (10) has straightforward interpretation: it is maximal when the changes in each step are equal to range (maximal possible change during one period), and is minimal when data are constant. The possible values vary from 0 to 1.

As a result, from (10) we can calculate the coefficients b_i which have the form of:

$$b_i = \frac{1-\xi(y_i)}{K}. \tag{11}$$

Proposed variability (volatility) measure can better capture the local nature of the time series volatility what is presented in Fig. 2.

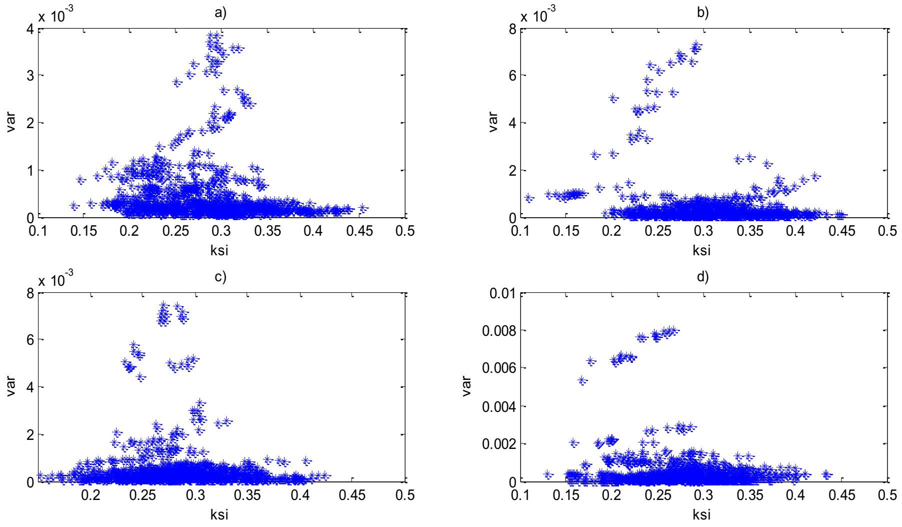


Fig. 2. Relations between the variance and *ksi* factor for: a) CAC40, b) Nikkei, c) BOVESPA, d) HANGSENG

It can be seen that the volatility of particular financial instruments for different time windows and different periods measured by (10) is distributed more evenly than measured by variance which tends to be grouped. Uniform distribution of volatility allows for better interpretation and is not without significance when learning separation system.

3 Practical Experiment

In this experiment we investigated the possibility to analyse relations among financial time series with the blind signal separation methods. In particular, the application included: common signals identification and Value at Risk computation for filtrated signals.

Therefore, in our experiment we could distinct following steps illustrated in Fig. 3:

1. Collect the original time series into multivariate variable **X**;
2. Decompose the matrix **X** with ICA into signals **S**;
3. Extract specific signals that may over-influence the data analysis;
4. Eliminate the signals and mix the rest with system inverse to decomposition;
5. Calculate VaR on filtered time series.

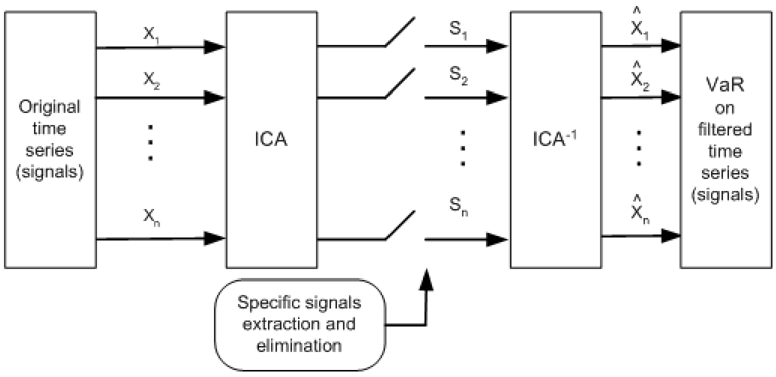

Original time series (signals)
x_1
x_2
x_n
ICA
s_1
s_2
s_n
ICA^{-1}
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_n$
VaR on filtered time series (signals)
Specific signals extraction and elimination

Fig. 3. The scheme for Independent Component Analysis and its application for VaR

Common signals were identified among 16 signals on logarithmic rates of return, based on 965 daily observations from 04-01-2008 until 01-08-2012 of: WIG, CAC40, DAX, Nikkei, BOVESPA, FTSE100, SP500, BRLPLN, EURPLN, GBPPLN, JPYPLN, USDPLN, HANGSENG, NASDAQ, BUX and HUFPLN, please see Fig. 4. The horizontal axis presents 16 loads, and the vertical axis shows the value of the load calculated as mixing matrix A multiplied by standard deviation of independent signals.

After decomposition of 16 time series representing various international economies and currencies we can obtain 16 independent signals. In Fig. 4 we can observe how big the load of particular independent component is in real series. The load of component s_j in series x_i is measured as $a_{ij}\sigma_j$, where a_{ij} is an element of mixing matrix and σ_j is a standard deviation of signal s_j.

For the index of Warsaw Stock Exchange (WIG) we can observe that it consists mainly from signals 7 and 14, where the signal 7 is important only to BUX

(Hungarian Stock Exchange) and the signal 14 is significant in CAC40, Bovespa, Hangseng, FTSE100, and somehow also Nikkei and SP500. It seems that Polish stock market follows two patterns. One is general condition of international markets and the second is emerging markets in Central and Eastern Europe.

We can also observe that all of the currencies against Polish zloty have significant load of 6th independent signal. It means that this component represents the state of Polish currency.

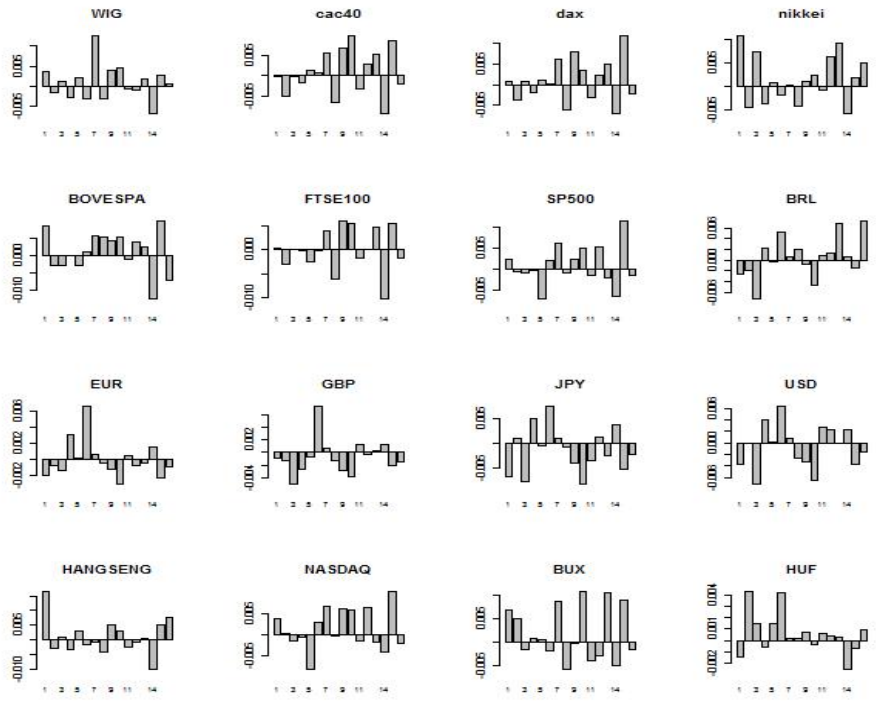


Fig. 4. Loads of independent signals in financial time series calculated as mixing matrix **A** multiplied by standard deviation of independent signals

In case of 1st signal it seems important for Asian stock markets (HangSeng and Nikkei), and also Japan yen (JPY) and Brazilian market (Bovespa).

In general, we can conclude that ICA methods enable identification of common signals that influence groups of markets, where the groups are constructed in concise way.

In the second part of the experiment we perform the analysis of Value at Risk 95. Not to be concentrated on technical issues regarding risk measure calculation like density estimation methods and autoregressive interdependencies we will approximate the VaR95 of logarithmic rates of return with 5th percentile calculated from historical data. We check how the filtration of source signals influences such risk measure. In Fig. 5 we show the experiment results with two lines: wide and narrow. The wide line shows changes in VaR95 if the signals with the growing load are filtered out. In particular, for each financial time series (e.g. WIG) we estimate the load of particular components, and filter out groups of signals starting from the lowest load. We can observe that filtering of the signals with the lowest loads does not influence VaR95, and only including within the group the higher loads gives significant change in risk measure – see convex wide line in Fig. 5. The narrow line shows changes in VaR95 if the signals with the decreasing load are filtered out. In particular, we can observe, that at the beginning the influence on VaR95 is significant and later on the line becomes flat – see concave narrow line in Fig. 5.

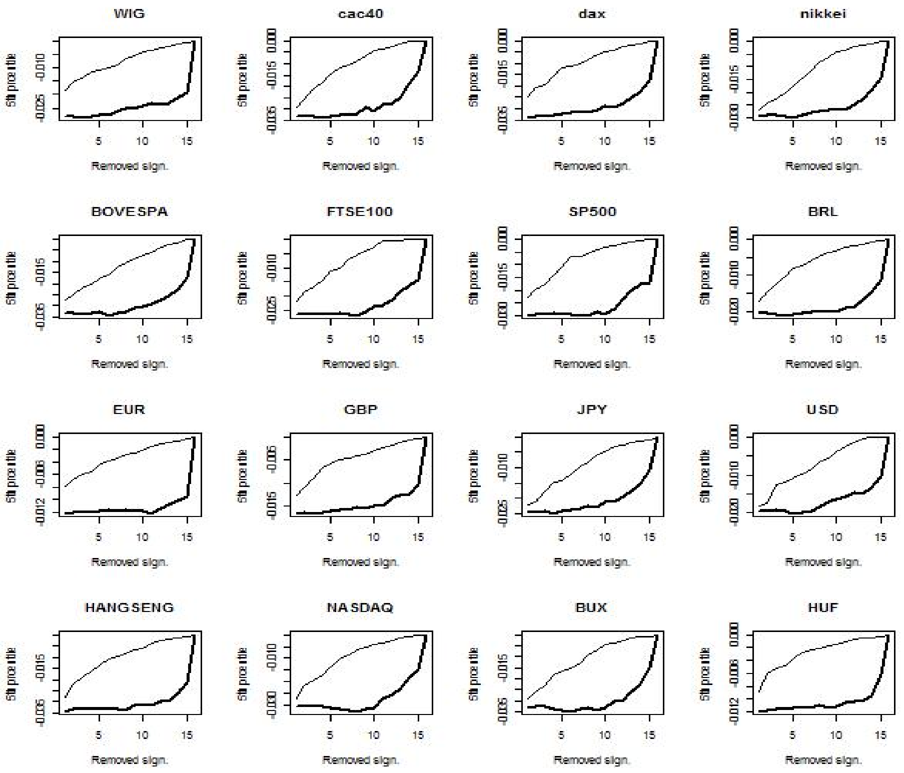


Fig. 5. VaR approximated by 5th percentile for ICA filtrated time series. Wide line represents level of VaR95 where the groups of signals with smallest load are removed. Narrow line - groups of signals with largest load are removed.

While the resulting lines are legibly convex in case of increasing signal load and concave in case of decreasing load we might conclude on the influence of resource signal filtration to the risk measures. In particular, if ICA estimated components of lower load (not significant for given economic time series, possible to be interpreted as non-informative or noise) are filtered out, then the VaR is not influenced. If the signals of significant load (informative components) are filtered out, then the risk measures are significantly decreased.

4 Conclusions

In this paper, we showed that in an environment characterized by non-linearities and non-Gaussianity, the ICA methodology can discover an underlying structure in financial time series for the purpose of risk measurement. In our approach, ICA is used to filter components with specific statistical properties enabling data interpretation. In particular, this refers to the issue of handling the impact of rare events. Rare events such as the sudden market breakdown cannot be regarded as a statistical outlier, since its impact has long-term consequences. Depends on the analysis context we can underline their occurrence or reduce their impact what introduce flexibility to VaR modelling. The experiment conducted on historical data confirmed the validity of this approach.

However, the characteristic of financial time series, in particular the nonstationarity in terms of higher order statistics (3rd, 4th), make it difficult to apply typical ICA algorithms right away and some modifications of algorithms are recommended. The local version of Natural Gradient ICA algorithm proposed in this work can be adjusted depending on the actual volatility of financial instruments, measured with author's variability measure. Along with the selection of non-linearity, based, for instance, on Extended Generalized Lambda Distribution, we get the algorithm exploring many important characteristics of time series such as variability (volatility), skewness, kurtosis what brings us toward better financial market behaviour understanding.

Acknowledgements. The work was funded by the National Science Center in Poland based on decision number DEC-2011/03/B/HS4/05092.

References

[1] Cardoso, J.F.: High-order contrasts for independent component analysis. Neural Computation 11(1), 157–192 (1999)

[2] Chen, Y., Hardle, W., Spokoiny, V.: Portfolio value at risk based on independent component analysis. Journal of Computational and Applied Mathematics 205(1) (2007)

[3] Cichocki, A., Amari, S.: Adaptive Blind Signal and Image Processing. John Wiley, Chichester (2002)

[4] Choi, S., Cichocki, A., Belouchrani, A., Second, A.: order nonstationary source separation. Journal of VLSI Signal Processing 32, 93–104 (2002)

[5] Georgiev, P., Cichocki, A.: Robust Independent Component Analysis via Time-Delayed Cumulant Functions. IEICE Trans. Fundamentals E86-A (3), 573–579 (2003)

[6] Holton, G.: Value-at-Risk. Theory and Practice Academic Press (2003)

[7] Hull, J.: Risk Management and Financial Institutions. John Wiley (2012)

[8] Hyvarinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. John Wiley, New York (2001)

[9] Jorion, P.: Value at Risk. McGraw-Hill (2006)

[10] Morgan, J.P.: Riskmetrics Technical Document, 3rd edn., New York (1995)

[11] Karian, Z.A., Zaven, A., Dudewicz, E.J.: Fitting statistical distributions: the Generalized Lambda Distribution and Generalized Bootstrap methods. Chapman & Hall (2000)

[12] Karvanen, J., Eriksson, J., Koivunen, V.: Adaptive Score Functions for Maximum Likelihood ICA. VLSI Signal Processing 32, 83–92 (2002)
[13] Molgedey, L., Schuster, G.: Separation of a mixture of independent signals using time delayed correlations. Physical Review Letters 72(23), 3634–3637 (1994)
[14] Müller, K.-R., Philips, P., Ziehe, A.: JADETD: Combining higher-order statistics and temporal information for blind source separation (with noise). In: Cardoso, J.F., Jutten, C., Loubaton, P. (eds.) ICA 1999, pp. 87–92 (1999)
[15] Pham, D.-T., Cardoso, J.-F.P.: Blind separation of instantaneous mixtures of nonstationary sources. IEEE Transactions on Signal Processing 49(9), 1837–1848 (2001)
[16] Szupiluk, R., Wojewnik, P., Ząbkowski, T.: Multiplicative ICA algorithm for interaction analysis in financial markets. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS, vol. 7268, pp. 608–615. Springer, Heidelberg (2012)
[17] Szupiluk, R., Wojewnik, P., Zabkowski, T.: Model Improvement by the Statistical Decomposition. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 1199–1204. Springer, Heidelberg (2004)
[18] Wu, E.H., Yu, P.L., Li, W.K.: Value at risk estimation using independent component analysis-generalized autoregressive conditional heteroscedasticity (ICA-GARCH) models. International Journal of Neural Systems 16(5), 371–382 (2006)

Wind Power Resource Estimation with Deep Neural Networks

Frank Sehnke[1], Achim Strunk[2], Martin Felder[1], Joris Brombach[2], Anton Kaifel[1], and Jon Meis[2]

[1] Zentrum für Sonnenenergie- und Wasserstoff-Forschung, Industriestr. 6, 70565 Stuttgart, Germany
[2] EWC Weather Consult GmbH, Schönfeldstrae 8, 76131 Karlsruhe, Germany

Abstract. The measure-correlate-predict technique is state-of-the-art for assessing the quality of a wind power resource based on long term numerical weather prediction systems. On-site wind speed measurements are correlated to meteorological reanalysis data, which represent the best historical estimate available for the atmospheric state. The different variants of MCP more or less correct the statistical main attributes by making the meteorological reanalyses bias and scaling free using the on-site measurements. However, by neglecting the higher order correlations none of the variants utilize the full potential of the measurements. We show that deep neural networks make use of these higher order correlations. Our implementation is tailored to the requirements of MCP in the context of wind resource assessment. We show the application of this method to a set of different locations and compare the results to a simple linear fit to the wind speed frequency distribution as well as to a standard linear regression MCP, that represents the state-of-the-art in industrial aerodynamics. The neural network based MCP outperforms both other methods with respect to correlation, root-mean-square error and the distance in the wind speed frequency distribution. Site assessment can be considered one of the most important steps developing a wind energy project. To this end, the approach described can be regarded as a novel, high-quality tool for reducing uncertainties in the long-term reference problem of on-site measurements.

1 Introduction

Due to the strong inter-annual variability of wind velocities in most regions that are interesting for wind power generation, reliable wind resource assessments require long-term time series for every specific location. Many approaches use data from meteorological reanalyses which represent the best historical estimate for the atmospheric state on regular grids. Moreover, these data sets allow for incorporating a sufficient length of the time series giving robust estimates in a climate context. Virtual time series at site level are produced by downscaling techniques, either using regional to local numerical model simulations including Computational Fluid Dynamics (CFD), or statistical or parametric methods,

© Springer-Verlag Berlin Heidelberg 2013

or combinations thereof. Finally these long-term time series are combined with local observations or measurements from representative locations nearby using a technique called Measure-Correlate-Predict (MCP).

MCP is described by [1] as: *"MCP methods model the relationship between wind data (speed and direction) measured at the target site, usually over a period of up to a year, and concurrent data at a nearby reference site. The model is then used with long-term data from the reference site to predict the long-term wind speed and direction distributions at the target site."*

In recent years, however, meteorological reanalyses from Numerical Weather Prediction (NWP) systems became increasingly available. The correlation between reanalysis data for the site and the measurements at the site are much higher and so MCP is now used with NWP reanalyses but with the same techniques described in [1]. These techniques range from standard deviation correction (to remove the overall scaling error) over wind speed ratio corrections (to remove the overall bias) to vector regression and linear regression – the latter being the standard and therefore heavily employed in industrial aerodynamics.

High quality MCPs enable reconstruction of the long-term historical wind resource using a limited number of observations while, at the same time, achieving reduced uncertainties. Due to the cost of gathering observations at site level and the need to have early, reliable estimates of future economic return, a state-of-the-art MCP and high-end long-term time series are key aspects during the initial phase of wind energy projects.

The standard MCP variants fail, however, to utilize high order correlations of the long-term historical wind resource data with the limited number of observations. We show that Deep Neural Networks (DNN) [2] apparently make use of these higher order correlations. The implementation is fully tailored to the requirements of MCP in the context of wind resource assessment. We show the application of our method to a set of different locations and compare the results to a simple linear fit to the wind speed frequency distribution as well as to a standard linear regression MCP. The neural network based MCP outperforms both other methods with respect to correlation, root-mean-square error and the distance in the wind speed frequency distribution. Site assessment can be considered one of the most important steps developing a wind energy project. To this end, the approach described can be regarded as a novel, high-quality tool for reducing uncertainties in the long-term reference problem of on-site measurements.

In the following sections we will introduce the long-term data under consideration as well as the observations used for this study. After a description of the DNN setup, different MCP variants will be introduced and the results will be compared to the method under consideration.

2 Method

2.1 Long-Term Data

EWC's Wind Potential Analysis employs the reanalysis product from the MERRA project [3], which covers the time range from 1979 to the present.

A parametric downscaling approach, which bases on [4], is applied to the hourly vertical wind profiles at a specific site. This approach takes into account, besides others, local orography inferred from high resolution digital elevation models. Using similarity theory, atmospheric stability and local vegetative roughness, individual hourly vertical wind profiles are calculated. The resulting wind velocities and directions then serve as artificial wind time series at (wind turbine) hub height. In this study the additional data sets of orography and land cover have a global resolution of roughly $1 \times 1\,\text{km}^2$. The parametric downscaling approach within EWC's Wind Potential Analysis is constantly being evaluated by numerous observational time series (e.g., [5]). In addition to resource assessment, this approach is also successfully applied for wind power forecast services [6].

2.2 Observational Data

In the current study we present the application of the novel MCP to a number of different locations. The hourly wind speed observations are either based on nacelle anemometers at hub height or have been gathered on net masts. The time periods of available measurement data are given in Table 1. The locations have been chosen

- due to their moderate skill in the initial long-term time series, and
- aiming at representing different terrain characteristics ranging from off-shore over near-shore flat to on-shore, complex and affected by forests.

Hub heights vary between about 60 m and 120 m above ground. In each of the experiments discussed below, about 10% of the data was withdrawn from training and used as test data.

Table 1. List of stations under consideration including their main characteristics. The last column gives the number of net months used for testing.

Station	Characteristics	Area	Height	Observations	Months test
1	on-shore, flat	Germany	61 m	2006/01 - 2009/06	4
2	offshore	Europe	69 m	2011/01 - 2012/05	2
3	on-shore, complex	Germany	100 m	2006/11 - 2009/06	3
4	near-shore, flat	USA	118 m	2006/09 - 2008/07	3
5	near-shore, complex	Greece	80 m	2008/05 - 2010/06	3
6	on-shore, forest	France	78 m	2006/11 - 2010/06	4

2.3 A Deep Neural Network Setup for MCP

The following DNN setup is used to build the MCP system.

1. The data is balanced to get an equal distribution over the targets, which are the wind speeds. This first step is crucial for the MCP procedure to *force* the DNN to conserve the target distribution. The wind speed distribution in this context has more impact on the final resource assessment than the root-mean-square error (RMSE) for wind speed, because the final resource assessment is the sum of wind speeds seen at the specific wind power plants.

2. Feature selection is applied that uses input neuron pruning based on weight strengths achieved under a L1 regularization [7]. Describing the exact feature selection mechanism is out of scope for this work. It is based mainly on [8] and identifies the problem relevant inputs that stem from the NWP.
3. The optimal network structure (depth of network and number of hidden neurons per layer) is identified using the RMSE on the test set as quality measure. Policy Gradients with Parameter-based Exploration (PGPE) [9,10] is used for this optimization. The optimal DNN architectures range from two hidden layers with 256 hidden neurons per layer up to four hidden layers with 512 hidden neurons, depending on the amount and quality of the data for the different locations.
4. Training under L2 regularization is performed with RPROP [11]. Pre-training with Restricted Boltzmann Machines (RBM) [12] was conducted for some locations but yielded no further improvement, so that we omit these experiments in the results section. Also, different weight initializations like the sparse initialization technique from [13,14] did not yield any improvement in convergence speed or final quality.

For all above steps we used the Learn-O-Matic framework [15] for fast Graphics Processing Unit (GPU) training, based on Cudamat [16] and Gnumpy [17]. This is also the reason for the number of hidden units per layer being restricted to multiples of 128 – it makes the computation on the GPU much faster.

3 Results

In order to evaluate the performance of the DNN approach under consideration ("NN-MCP") different additional experiments have been carried out. First, the bias has been removed from the original long-term time series over the training period ("bias-free"). Additionally, a linear fit has been done targeting on the best possible fit to the observed wind speed frequency distribution ("min. dist. error"). Finally, a standard, linear regression MCP has been employed based on 4 stability classes and 8 wind direction sectors as is the standard in the field ("standard MCP"). Figure 1 summarizes the results in terms of main skill scores: bias error, RMSE, correlation coefficient and the skill of the wind speed frequency distribution. Compared to the original long-term time series, which shows partly significant biases and for which these stations have been chosen for this study, all other methods clearly reduce errors on the test data set. While already providing high correlation coefficients at an hourly level, especially the NN-MCP method is able to increase correlation and thus further reduce the RMSE. This fact is additionally illustrated in Figure 2 which shows the scatter diagrams for stations 5 and 6 for three different methods: the linear fit to the frequency distribution ("min. dist. error"), the standard MCP and the NN-MCP. Especially for low and high wind speeds the NN-MCP is able to better reproduce the observed values leading to significantly increased correlation coefficients.

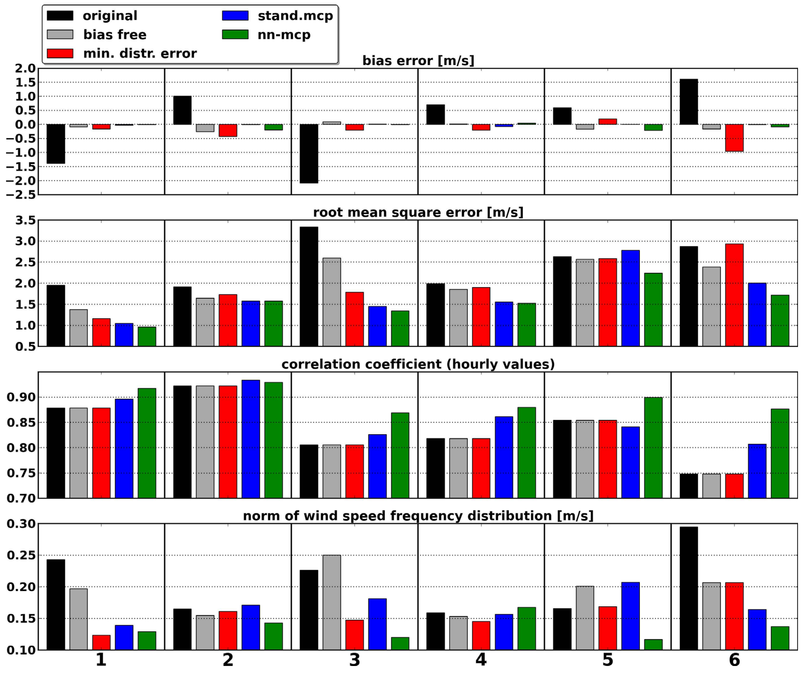


Fig. 1. Error scores (bias, root-mean-square error, correlation coefficient and error in the wind speed frequency distribution) on the test data for the original time series, a bias free version of the original data, a linear fit to the frequency distribution, a standard linear MCP method as well as for the neural network MCP

Evaluating generalization errors of the wind speed frequency distribution and by that the effect on the expected energy yield, the L2-norm of the wind speed frequency distribution error is also given for all methods and stations in Figure 1. Except for station 4, the NN-MCP clearly outperforms the standard linear MCP, which partly yield worse results than the simple linear fit of the frequency distribution. In order to stress this fact, Figure 3 shows histograms of the wind speed frequency distribution of the test data for the linear fit to the distribution ("min. dist. error"), the"standard MCP" and the NN-MCP. In addition to the general misfit discussed above, these graphs show the main characteristics of the approaches. For station 5, the shape of the frequency distribution is already quite well captured by the original data set, such that the linear fit leads to satisfying results. This healthy initial state is then deteriorated by the standard MCP, which fits each of the sector-stability combinations separately and therefore modulates the distribution. In contrast, the NN-MCP method strongly improves the wind speed frequency distribution on the test data set. For station 6 the initial distribution is of moderate quality and the standard MCP successfully limits the overestimation of high wind speed occurrences above 15 $\frac{\mathrm{m}}{\mathrm{s}}$. This leads to a satisfying frequency distribution between 5 $\frac{\mathrm{m}}{\mathrm{s}}$ and 15 $\frac{\mathrm{m}}{\mathrm{s}}$, while the number of small wind speeds remains underestimated. The NN-MCP fully reproduces the

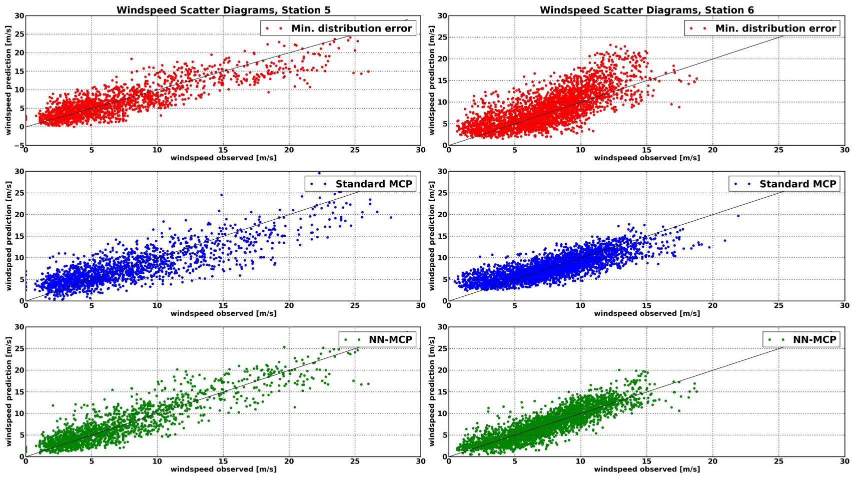


Fig. 2. Wind speed scatter diagrams for the linear fit to the frequency distribution (top), the standard MCP method (middle) and the neural network based MCP (bottom). Diagrams are given for station 5 (left panel) and station 6 (right panel).

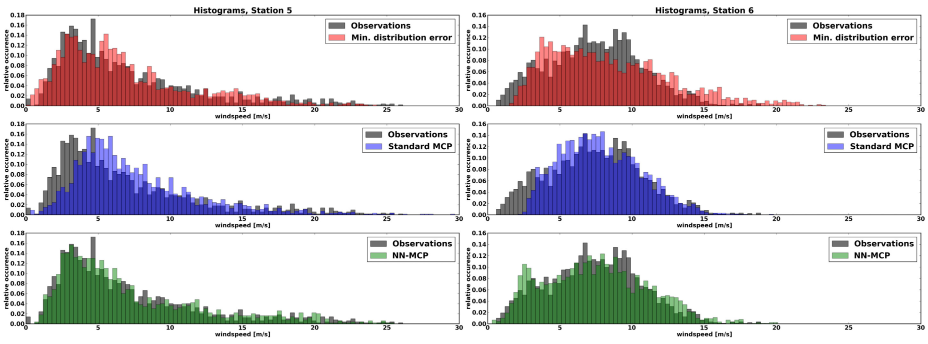


Fig. 3. Histograms of the wind speed frequency distribution for the linear fit (top), the standard MCP method (middle) and the neural network based MCP (bottom). Diagrams are given for station 5 (left panel) and station 6 (right panel).

frequency distribution for the test data set, leading to the small general misfit mentioned above.

In order to assess the skill of the expected energy production with respect to the methods under consideration, hourly wind speed values have been converted to hourly production data by using a single power curve for each of the stations. In this study we used a Nordex N117 (2.4 MW) turbine at constant air density. Results of the conversion are given in Table 2. The numbers stated make use of the test set periods. Annual production estimates delivered by the different techniques are compared to the observed ones calculated from hourly values. Masking errors in the frequency distribution below cut-in speed and above full-load speed, the errors in the wind speed frequency distribution do not directly translate into errors in the energy yield. However, the NN-MCP gives the lowest misfits in the annual production estimates. Averaging the absolute values of relative differences for all stations shows the superior quality of the NN-MCP compared to the standard MCP method in reconstructing wind speed frequency distribution and annual production estimates.

Table 2. Comparison of the mean annual energy yields [GWh] ("Obs.") wrt. the methods employed. Differences to observed annual totals are given in per cent.

Station	Obs.	Min. dist. error	Std. MCP	NN MCP
1	5.00	5.35 +6.9%	4.64 -7.1%	**5.10 +2.0%**
2	14.53	14.99 +3.1%	14.92 +2.7%	**14.71 +1.2%**
3	5.83	6.34 +8.6%	5.24 -10.2%	**5.75 -1.4%**
4	10.25	10.52 +2.6%	10.07 -1.8%	**10.09 -1.6%**
5	7.81	7.96 +2.0%	8.36 +7.1%	**7.91 +1.3%**
6	11.00	11.61 +5.6%	10.82 -1.6%	**10.83 -1.5%**
mean abs misfit:		4.8%	5.08%	**1.5%**

4 Conclusions and Future Work

We have shown that the developments in neural computation of the last years, namely deep neural networks, are well suited to cope with the requirements of high quality wind power resource estimation. The only somewhat uncommon preprocessing step for this data was a balancing of the target frequency distribution.

Curiously, pre-training the deep structures with RBMs or greedy layer-wise training, both of which have been found advantageous in the literature, yield no advantage in this scenario despite the networks containing up to 4 hidden layers. Likewise, sparse initialization and similar tricks of the trade led to no further improvement.

The results obtained are far superior to what can be achieved by state-of-the-art methods from the field of industrial aerodynamics. The system presented will be available as a commercial product in the near future and hopefully help the integration of renewable energies into the grid.

We see interesting future work in using ensembles of DNNs and dropout [18] to gain additional improvements by using multiple models.

References

1. Rogers, A.L., Rogers, J.W., Manwell, J.F.: Comparison of the performance of four measure–correlate–predict algorithms. Journal of Wind Engineering and Industrial Aerodynamics 93(3), 243–264 (2005)
2. Bengio, Y.: Learning deep architectures for ai. Foundations and Trends® in Machine Learning 2(1), 1–127 (2009)

3. Rienecker, M.M., Suarez, M.J., Gelaro, R., Todling, R., Bacmeister, J., Liu, E., Bosilovich, M.G., Schubert, S.D., Takacs, L., Kim, G.K., et al.: MERRA: NASA's modern-era retrospective analysis for research and applications. Journal of Climate 24(14), 3624–3648 (2011)
4. Howard, T., Clark, P.: Correction and downscaling of NWP wind speed forecasts. Meteorological Applications 14(2), 105–116 (2007)
5. Meis, J., Kuntze, K.: Wind potential analysis for great heights with archived GFS data. In: European Wind Energy Conference and Exhibition, vol. PO. ID 173 (2010)
6. Sack, J., Strunk, A., Meis, J., Sehnke, F., Felder, M.D., Kaifel, A.K.: From ensembles to probabilistic wind power forecasts - how crucial is the ensemble size? In: Proceedings of the 2nd International Conference on Energy and Meteorology, ICEM, Toulouse (2013)
7. Ng, A.Y.: Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 78. ACM (2004)
8. Felder, M.D., Sehnke, F., Kaifel, A.K.: Automatic feature selection for combined iasi/gome-2 ozone profile retrieval. In: Proceedings of the 2012 EUMETSAT Meteorological Satellite Conference (2012)
9. Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., Schmidhuber, J.: Parameter-exploring policy gradients. Neural Networks 23(4), 551–559 (2010)
10. Sehnke, F., Graves, A., Osendorfer, C., Schmidhuber, J.: Multimodal parameter-exploring policy gradients. In: 2010 Ninth International Conference on Machine Learning and Applications, ICMLA, pp. 113–118. IEEE (2010)
11. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: IEEE International Conference on Neural Networks 1993, pp. 586–591. IEEE (1993)
12. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: ACM International Conference Proceeding Series, vol. 227, pp. 791–798 (2007)
13. Martens, J.: Deep learning via hessian-free optimization. In: Proceedings of the 27th International Conference on Machine Learning, ICML, vol. 951 (2010)
14. Sutskever, I.: Training Recurrent Neural Networks. PhD thesis, University of Toronto (2013)
15. Sehnke, F., Felder, M.D., Kaifel, A.K.: Learn-o-matic: A fully automated machine learning suite for profile retrieval applications. In: Proceedings of the 2012 EUMETSAT Meteorological Satellite Conference (2012)
16. Mnih, V.: Cudamat: a cuda-based matrix class for python. Department of Computer Science, University of Toronto. Tech. Rep. UTML TR 4 (2009)
17. Tieleman, T.: Gnumpy: an easy way to use gpu boards in python. Department of Computer Science, University of Toronto (2010)
18. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)

Wavelet Neural Networks for Electricity Load Forecasting – Dealing with Border Distortion and Shift Invariance

Mashud Rana and Irena Koprinska

School of Information Technologies, University of Sydney,
Sydney, Australia
{mashud,irena}@it.usyd.edu.au

Abstract. We consider a wavelet neural network approach for electricity load prediction. The wavelet transform is used to decompose the load into different frequency components that are predicted separately using neural networks. We firstly propose a new approach for signal extension which minimizes the border distortion when decomposing the data, outperforming three standard methods. We also compare the performance of the standard wavelet transform, which is shift variant, with a non-decimated transform, which is shift invariant. Our results show that the use of shift invariant transform considerably improves the prediction accuracy. In addition to wavelet neural network, we also present the results of wavelet linear regression, wavelet model trees and a number of baselines. Our evaluation uses two years of Australian electricity data.

Keywords: electricity load forecasting, wavelet-based neural networks, shift invariance, non-decimated wavelet transform, border distortion, linear regression, model trees.

1 Introduction

Predicting future electricity loads (demands) from previous loads is required for the efficient management of electricity networks. This includes optimal scheduling of electricity generators, setting the minimum reserve, planning maintenance and supporting the transactions of electricity companies in competitive electricity markets. Accurate forecasting is needed to ensure reliable supply while keeping the costs low.

A number of load forecasting methods have been developed. The most prominent methods include exponential smoothing and ARIMA [1], neural networks and linear regression [2,3]. Recently, wavelet-based decomposition has been used in conjunction with neural networks [4-7] and shown to produce highly accurate results. The key idea is to decompose the load into different frequency components and predict them separately. However, there are two main issues that have received very little attention in previous work: 1) shift variance of the wavelet transform and 2) border distortion when decomposing the signal.

To the best of our knowledge all previous wavelet neural network methods except [7] use the standard Discrete Wavelet Transform (DWT). DWT has an important

© Springer-Verlag Berlin Heidelberg 2013

drawback – it is shift variant. This means that the DWT of a signal and the DWT of a shifted version of this signal will be different for the same points included in the signal and its shifted version. This is a potential problem for forecasting applications using a sliding window, such as electricity load forecasting, as the wavelet coefficients of the same points included in different sliding windows will be different, which may affect the pattern extraction and reduce prediction accuracy. There is no previous study assessing the impact of the shift variance/invariance in electricity load forecasting.

Real-time applications of DWT also require extending the signal at the boundaries of the sliding window to reduce the border distortion during decomposition. With the exception of [4], this problem hasn't been dealt with in previous work.

In this paper, we consider wavelet-based neural networks for electricity load forecasting and investigate two problems: 1) dealing with border distortion during the wavelet decomposition and 2) the importance of using a shift invariant wavelet transform. Firstly, in order to minimize border distortion we propose a new method for signal extension that uses a machine learning technique. It shows superior performance when compared with three standard methods. Secondly, we compare the performance of shift-variant and shift-invariant wavelet approaches. To assess better this impact, we consider not only wavelet neural networks, but also wavelet linear regression and wavelet model trees. Our results show that the use of shift invariant wavelet transform improves the accuracy in all cases.

2 Wavelet Transform and Shift Invariance

The wavelet transform decomposes the signal into a hierarchical set of low frequency components called *approximations* and high frequency components called *details*. The approximations represent the general trend of the signal while the details represent the difference between two successive approximations. When the signal is observed at discrete times, as in the electricity load data, we can apply the DWT. The Mallat's implementation of DWT [8] is widely used; it is based on filtering and is computationally efficient.

The standard DWT is not shift invariant. The DWT of a signal is shift invariant if the DWT coefficients of the signal values do not depend on the origin of the transformation. Time series applications such as electricity load prediction use a sliding window. Shift invariance is a desired property for them as without it the wavelet coefficients of the same point will be different in different sliding windows. This may affect the ability to recognize patterns in data and, hence, decrease the prediction accuracy.

The DWT can be made shift invariant by removing the decimation step in the wavelet decomposition. There are several algorithms to do this; we used the *epsilon-decimated algorithm* [9]. The key idea is to compute and retain both the odd and even coefficients at each decomposition level resulting in no loss of wavelet coefficients. The resulting shift invariant DWT is called Non-Decimated Discrete Wavelet Transform (NDWT).

3 Data and Problem Statement

We consider 5-minute electricity load data for the state of New South Wales in Australia, for the years 2006 and 2007, provided by Australian Energy Market Operator [10]. Given a time series containing n observations $X_1, X_2, \ldots, X_n$, our goal is to forecast X_{n+1}, the value of the series one step ahead, using the data up to time n. The 2006 data was used as *training data* (for feature selection and building prediction models) and the 2007 data was used as *testing data* (for evaluating the performance of these models). The training and testing sets contain 105,120 samples each.

4 Wavelet-Based Load Forecasting

Our wavelet-based approach for load forecasting consists of two main steps: 1) load decomposition into several components, 2) conducting feature selection and building a prediction model for each component, and then combining the individual predictions into a final prediction.

4.1 Wavelet Decomposition of the Electricity Load

We apply both the classical DWT and the NDWT for multilevel decomposition of the original electricity load data. Daubechies wavelet of order 4 (DB4) has been chosen over other wavelet types as it is sufficiently regular and smooth for our dataset. After an empirical evaluation of different decomposition levels we chose to decompose the data up to level 3. This resulted in four components: A_3, D_3, D_2 and D_1.

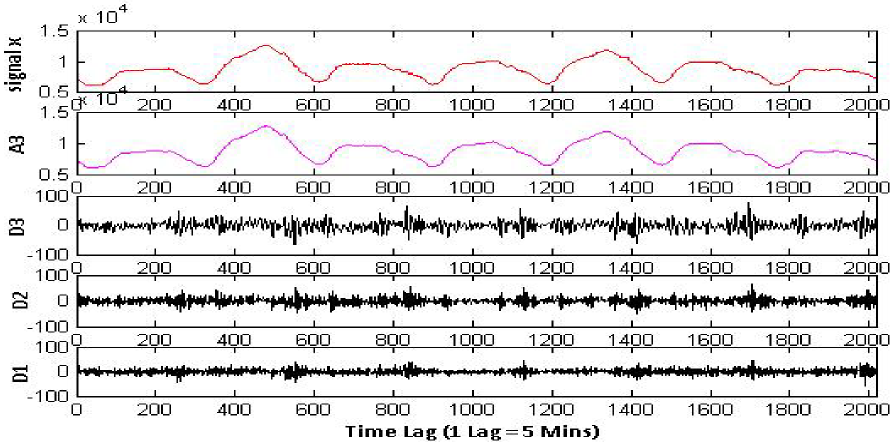


Fig. 1. Electricity load series using NDWT

Fig. 1 shows these components for a subset of our data using NDWT. We can see that the low frequency component A_3 is a smoother version of the original signal. The high frequency component D_3 also contains useful information, e.g. relatively high values after the peaks in A_3. D_2 and D_1 are similar and show more irregular fluctuations that are most probably due to random variations in the load caused by fluctuations in residential load, weather changes and measurement errors.

The wavelet decomposition of *training data* is done offline for each training point using a sliding window of size *L*. The decomposition of the *testing data* is done online, for each testing point, again using a sliding window of size *L*. The value of *L* is *p+f*, where *p* is the number of the previous points (3 previous weeks in our case, 3*2016=6048 points) and *f* is the number of future points needed to avoid border distortion during decomposition (see the next section). These future points are predicted before the wavelet decomposition is done using the method described in the next section. The value of *f* depends on the decomposition level and is defined as: $f=flen*2^{l-1}-1$, where: *flen* is the filter length, *flen=8* for our case as we use DB4; *l* is the decomposition level and varies from 1 to 3 in our case. Hence, the number of future points *f* and the length of the sliding window are: *f*=7 and *L*=6055 for *l*=1, *f*=17 and *L*=6165 for *l*=2 and *f*=31 and *L*= 6079 for *l*=3.

4.2 Dealing with Border Distortion

Before decomposing the signal, it is necessary to apply a method for reducing the border distortion. Border distortion is one of the major problems in wavelet decomposition of a finite length signal. It includes both distortion on the left and right sides of the signal. In our task the distortion at the left side (least recent values) can be avoided completely by using an extended window of previous values. The distortion at the right side (most recent values) cannot be avoided as it is not possible to use future values. Hence, our goal is to minimize the distortion at the right side by using an appropriate method for signal extension.

The standard methods for dealing with border distortion extend the signal by padding extra points at the boundaries. There are three main methods: 1) symmetric extension (*sym*) which mirrors the points next to the boundary; 2) smooth padding (*spd*) which extrapolates the signal using the first derivatives of the edges and 3) periodic extension (*ppd*) which extends the signal periodically starting from the beginning. These methods introduce discontinuity at the signal boundaries and may lead to incorrect estimation of the wavelet coefficients.

We propose a new method for signal extension. It appends previous actual load values at the left side (which are available in our task) and *predicted* load values at the right side. To compute the predicted load values, we apply the following method:

1) Feature selection – Using the whole training data, we apply autocorrelation analysis to select important lag variables. Autocorrelation is a popular and suitable approach as the load data is highly linearly correlated. We first identify the 7 highest autocorrelation spikes which were at: 1) lag 1 (previous lag), 2) lag 2016 (1 week ago), 3) lag 288 (1 day ago), 4) lag 4032 (2 weeks ago), 5) lag 1728 (6 days ago), 6) lag 6048 (3 weeks ago) and 7) lag 2304 (8 days ago). We then extract the lag variable at the spike and its two neighbors (before and after). The first spike (at lag 1) is an exception as there are no lag variables after it and also because it corresponds to the strongest linear dependence; we extract its lag variable and the previous 10 lag variables. This results in 1*11+6*3=29 extracted variables in total.

2) Building a prediction model – Using the selected variables and the training data, we build a prediction model to forecast the load value one step ahead. We chose Linear Regression (LR) as the prediction model. Our previous work [11] has shown that it is an accurate and fast prediction method.
3) Forecasting *f* future load values to extend the signal on the right side – The trained prediction model in the previous step is used to forecast *f* future values (*f*=7, 17 or 31 for decomposition levels 1, 2 and 3, respectively). We use an iterative method: e.g. the value X_{t+1} is predicted and appended, then this predicted value is used to predict X_{t+2}, etc.

Fig. 2 compares our method for signal extension with the three standard methods and the actual signal (ground truth, not available in practice). It presents the results for the low frequency component A_3 for NDWT. We can see that all standard methods show substantial deviations from the actual signal – *ppd* is least accurate, followed by *sym* and *spd*. In contrast, our method is able to follow the actual signal very closely. Hence, our method completely avoids the distortion at the left border and significantly reduces the distortion at the right border.

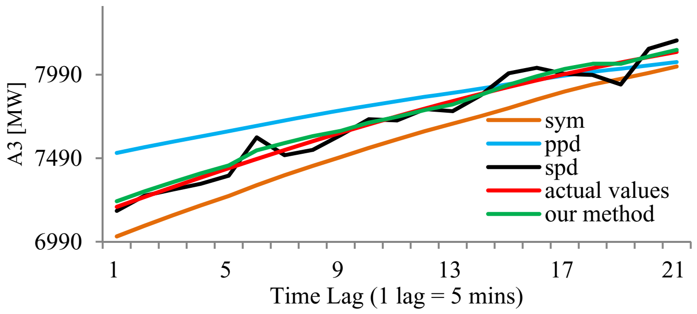


Fig. 2. Comparison of signal extension methods

4.3 Feature Selection and Wavelet Prediction Models

Good feature selection is essential for accurate prediction. We compute the autocorrelation of each wavelet component and use it to select features for this component. The autocorrelation function shows the linear correlation of the time series with itself at different lags; spikes close to 1 and -1 correspond to strong dependencies, and hence informative variables. We extract lag variables from the areas of the 7 highest spikes. As the highest spike is the most important, we extract more variables from its neighborhood. Likewise, as A_3 is the most important wavelet component, we extract more variables from it. The extracted features are:

- for A_3 - the lag variables of the 7 highest spikes and their two neighbors (before and after), except for the highest spike (at lag 1) where we extract the 10 previous lag variables: 1*11+6*3=29 variables.
- for D_3, D_2 and D_1 - the lag variables of the 7 highest spikes and their two neighbors, except for the highest spike, where we extract the 5 previous lag variables: 1*6+6*3=24 variables.

The extracted features for each wavelet component are shown in Table 1.

Table 1. Selected features

Series	Selected variables to predict X_{t+1}	# Features (n)
A_3	X_{t-10} to X_t; XD_{t-1} to XD_{t+1}; $XD6_{t-1}$ to $XD6_{t+1}$; XW_{t-1} to XW_{t+1}; $XD8_{t-1}$ to $XD8_{t+1}$; $XW2_{t-1}$ to $XW2_{t+1}$; $XW3_{t-1}$ to $XW3_{t+1}$	29
D_3	X_{t-5} to X_t; XD_{t-1} to XD_{t+1}; $XD2_{t-1}$ to $XD2_{t+1}$; $XD6_{t-1}$ to $XD6_{t+1}$; XW_t, to XW_{t+1}; $XD8_{t-1}$ to $XD8_{t+1}$; $XW2_t$, to $XW2_{t+1}$	24
D_2	As D_3	24
D_1	X_{t-5} to X_t; XD_{t-1} to XD_{t+1}; $XD2_{t-1}$ to $XD2_{t+1}$; $XD5_{t-1}$ to $XD5_{t+1}$; $XD6_{t-1}$ to $XD6_{t+1}$; XW_t, to XW_{t+1}; $XD8_{t-1}$ to $XD8_{t+1}$	24

X_t - load on forecast day at time t; XD_t, $XD2_t$, $XD5_t$, $XD6_t$ $XD8_t$ - loads 1, 2, 5, 6 and 8 days before the forecast day at time t; XW_t, $XW2_t$, $XW3_t$- loads 1, 2 and 3 weeks before the forecast day at time t.

For each wavelet component we build a separate prediction model using the selected features. The final prediction is generated by combining the individual predictions with the inverse wavelet transform. Most of the existing wavelet-based approaches for load forecasting use Backpropagation Neural Network (BPNN) as the prediction algorithm. To better assess the impact of the shift invariant wavelet transform, in addition to BPNN, we also use LR and Model Tree Regression (MTR). While BPNN and LR are well known, MTR [12] is a newer method. It generates a decision tree for regression tasks that is converted into a set of rules. MTR typically produces a small set of rules that are easily understandable by humans, which is an advantage over the LR and BPNN prediction methods.

5 Results and Discussion

To assess the impact of the shift invariant property of the wavelet transform, we compare the results of our wavelet-based method for load forecasting using the standard DWT and also the NDWT. Table 2 presents the predictive accuracy results for these two cases, for the three different prediction algorithms.

To measure the predictive accuracy, we use two standard measures: *Mean Absolute Error (MAE)* and *Mean Absolute Percentage Error (MAPE)* defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|L\_actual_i - L\_forecast_i\right|, \; MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{L\_actual_i - L\_forecast_i}{L\_actual_i}\right| 100\,[\%]$$

where L_actual_i and $L\_forecast_i$ are the actual and forecasted electricity loads at the 5-minute lag i and n is the total number of predicted loads.

From Table 2 we can see that the use of the NDWT for decomposition of electricity load resulted in considerably higher accuracy than the use of DWT, for all three prediction algorithms. The improvements in terms of MAE and MAPE, respectively, are: 8.74% and 8.86% for BPNN, 14.40% and 14.16% for LR and 15.04% and 15.54% for MTR. All improvements are statistically significant at $p<=0.001$. These improvements are due to the shift invariant property of the NDWT.

In NDWT the wavelet coefficients for each data point remain the same for different sliding windows which helps to extract patterns in data, reduce noise and build more accurate prediction models.

Table 2. Comparison of DWT with NDWT

Pred.	DWT		NDWT	
algorithm	MAE [MW]	MAPE [%]	MAE [MW]	MAPE [%]
BPNN	27.70	0.316	25.28	0.288
LR	29.17	0.332	24.97	0.285
MTR	29.73	0.337	25.26	0.288

We also compare the wavelet-based methods with a number of non-wavelet baselines. The *industry model* is a typical prediction model used by industry forecasters [3]. It uses 11 variables (previous loads from the same day and 7 days ago), combined into 9 features with logarithmic transformation and differencing of successive values, and BPNN as prediction algorithm. $Naive_{mean}$ simply predicts the mean of the class variable in the training data. $Naive_{plag}$ predicts the load from the previous lag. $Naive_{pday}$ predicts the load from the previous day at the same time and $Naive_{pweek}$ predicts the load from the previous week at the same time.

Table 3. Comparison with baselines

Pred. models	MAE [MW]	MAPE [%]
Industry	27.58	0.314
$\text{Naive}_{\text{mean}}$	1159.42	13.484
$\text{Naive}_{\text{plag}}$	41.24	0.473
$\text{Naive}_{\text{pday}}$	453.88	5.046
$\text{Naive}_{\text{pweek}}$	451.03	4.940

The results for the baselines are shown in Table 3. All wavelet based methods, regardless of the type of wavelet transform (NDWT or DWT), outperformed all naïve baselines. NDWT also outperformed the industry model when used with any of the three prediction algorithms. However, DWT was outperformed by the industry model (similar accuracy for BPNN and lower for LR and MTR). The pairwise differences in accuracy between all wavelet-based models and all baselines are statistically significant at $p<=0.001$ except the difference between DWT with BPNN and the industry model. This comparison shows again the importance of NDWT for building accurate prediction models. Overall, the most accurate model was LR with NDWT achieving MAPE of 0.285%.

6 Conclusion

In this paper we considered the problem of electricity load forecasting using a wavelet-based approach. We studied two key issues: 1) how to deal with border distortion and 2) the importance of using a shift invariant transform. Our study was conducted using Australian electricity data for two years. Firstly, we proposed a new method for dealing with border distortion. It completely eliminates the distortion at the left side of the signal and shows minimal distortion on the right side in comparison to three standard methods. Secondly, we compared the performance of shift variant and shift invariant transforms in a wavelet-based prediction approach, using three different prediction algorithms: BPNN, LR and MTR. Our results showed that the application of NDWT (a shift invariant wavelet transform) resulted in considerably higher predictive accuracy than the use of DWT (a standard shift variant wavelet transform), for all three prediction algorithms. Hence, we recommend the application of NDWT for wavelet-based prediction of electricity load data.

Acknowledgement. Mashud Rana is supported by an Endeavour award and a NICTA scholarship.

References

1. Taylor, J.W.: An Evaluation of Methods for Very Short-Term Load Forecasting Using Minite-by-Minute British Data. International Journal of Forecasting 24, 645–658 (2008)
2. Charytoniuk, W., Chen, M.-S.: Very Short-Term Load Forecasting Using Artificial Neutal Networks. IEEE Transactions on Power Systems 15, 263–268 (2000)
3. Koprinska, I., Rana, M., Agelidis, V.G.: Yearly and Seasonal Models for Electricity Load Forecasting. In: International Joint Conference on Neural Networks (IJCNN), San Jose, pp. 1474–1481. IEEE Press (2011)
4. Reis, A.J.R., Alvis, A.P., da Silva, P.A.: Feature Extraction via Multiresolution Analysis for Short-Term Load Forecasting. IEEE Transactions on Power Systems 20, 189–198 (2005)
5. Chen, Y., Luh, P.B., Guan, C., Zhao, Y., Michel, L.D., Coolbeth, M.A.: Short-Term Load Forecasting: Similar Day-based Wavelet Neural Network. IEEE Transactions on Power Systems 25, 322–330 (2010)
6. Bashir, A.A., El-Hawary, M.E.: Applying Wavelets to Short-Term Load Forecasting Using PSO-Based Neural Network. IEEE Transactions on Power Systems 24, 20–27 (2009)
7. Zhang, B.-L., Dong, Z.-Y.: An Adaptive Neural-Wavelet Model for Short Term Load Forecasting. Electric Power Systems Research 59, 121–129 (2001)
8. Mallat, S.: A Theory for Multiresolution Signal Decomposition: the Wavelet Representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 11, 674–693 (1989)
9. Nason, G.P., Silverman, B.W.: The Stationary Wavelet Transform and Some Statistical Applications. In: Lecture Notes in Statistics, pp. 281–300 (1995)
10. Australian Energy Market Operator (AEMO), `http://www.aemo.com.au`
11. Koprinska, I., Rana, M., Agelidis, V.G.: Electricity Load Forecasting: A Weekday-Based Approach. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 33–41. Springer, Heidelberg (2012)
12. Holmes, G., Hall, M., Frank, E.: Generating Rule Sets from Model Trees. In: Foo, N.Y. (ed.) AI 1999. LNCS, vol. 1747, pp. 1–12. Springer, Heidelberg (1999)

Interactive Two-Level WEBSOM for Organizational Exploration

Timo Honkela[1] and Michael Knapek[1,2]

[1] School of Science, Department of Information and Computer Science
[2] School of Arts, Design and Architecture, Department of Media
Aalto University, P.O. Box 15400, FI-00076 Aalto, Finland

Abstract. Among the large number of applications of the self-organizing map (SOM) algorithm, creating maps of document collections have become commonplace since the introduction of the WEBSOM system. This article presents a novel development in WEBSOM research. The Interactive Two-Level WEBSOM, I2WEBSOM, includes two main components, a map of terms, and a dynamic map of documents. The map of terms is used to enable interactive feature selection and weighting. The map of documents is calculated using terminology-based feature vectors where their weights can be changed using the first-level map. In the experimental part, we focus on the application of creating maps of people based on their interest or competence profiles.

1 Introduction

We will describe in the following the classical WEBSOM method for information visualization, and how maps of people can be created using WEBSOM, for instance, to facilitate competence management.

1.1 WEBSOM in Exploration of Text Collections

The WEBSOM method was developed to facilitate interactive visual exploration of large document collections [1]. A central component of the WEBSOM is the self-organizing map (SOM) [2] that has proven to be an efficient and reliable means for projecting high-dimensional data into a low-dimensional space. The map consists of a number of model vectors $m_i(t)$ that are organized typically into a regular two-dimensional grid. For input data item $x(t)$ updated value $m_i(t+1)$ is computed iteratively using the well known update rule $m_i(t+1) = m_i(t) + (t)h_{ci}(t)[x(t)m_i(t)]$, where (t) is a scalar factor that defines the size of the update, index i is the model vector under processing, c is the index of the model that has the smallest distance from $x(t)$, and the factor $h_{ci}(t)$ is a smoothing kernel, also called the neighborhood function [2]. It has been shown that the SOM is a viable alternative to more recently developed methods that are based on information-theoretical or probability-theory principles especially when the trustworthiness of the visualization is used as the quality criterion [3].

The number of applications in analyzing and visualizing numerical data was already substantial by the beginning of 1990s even though the popularity of the SOM has since

© Springer-Verlag Berlin Heidelberg 2013

then grown to cover most areas of science and technology. The second important enabling step that led into the development of the WEBSOM method was the advent of the self-organizing semantic maps or, in other words, maps of words [4]. It was shown that the SOM can be used to create meaningful analysis of individual words based on the contextual statistics of each word. The relative syntactic or semantic similarity of two words can be detected by comparing the patterns of sentential contexts in which the words appear in text. The more similar the context patterns, the closer the relationship between the words is. This basic idea has been known already for a long time [5] and has been applied extensively during the recent years [6]. In the seminal work, artificially generated sentences were used [4]. The first map of words in written texts was based on analyzing the English translations of the fairy tales by Grimm brothers [7].

The idea of maps of documents had been brought up in early 1990s [8] and the first published experiments were based on the titles of documents [9]. The development of the WEBSOM method started in 1995 with the idea of creating "maps on information highways". The underlying motivation was based on the experiences in building a natural language database interface using traditional natural language processing and knowledge representation methods [10] and in applying developed modules in information retrieval [11]. A straightforward approach for creating maps of documents is to take the words appearing in the documents (or usually a subset of them) and to create a vector space in which each word corresponds to a dimension.

The first WEBSOM architecture had two levels: map of words was used to model similarities between words so that each word would not give rise to a single dimension but semantically related words could be grouped together [1]. The same kind of motivation is widely used in information retrieval applications that apply latent semantic analysis (LSA) [12].

In the two-level WEBSOM architecture, each document is encoded based on a map of words. A histogram of of the words in the document on the map is formed, and the histogram is normalized. This histogram resembles the vectors used in the vector space model but in this case the components of the vectors correspond to groups of words instead of single words [13].

In the later developments of the WEBSOM method, the two-level architecture was abandoned partly due to computational complexity issues. With the simplified architecture it was possible to create a map of millions of patents abstracts in which the number of connections between input and output layers was in the order of 10^{10} [14]. Since then, the WEBSOM concept has been applied in a large body of research (see, e.g., [15,16,17]).

1.2 Maps of People

One early application of the WEBSOM was creation of maps of people. In WSOM'97 conference, the participants were mapped based on the contents of their abstracts [18]. The idea has been extended in competence analysis [19] which is a central task in the area of human resource management (HRM). In HRM, data mining methods are becoming increasingly popular [20].

In the following, a novel development in WEBSOM research is described. The two-level architecture is re-introduced but based on a different approach in which the map of words is used for feature selection and weighting. Moreover, the user interface is built to visualize also the organization process, not only the end result. This visualization is based on another two-level process in which the map units are adapted according to the SOM learning rule and the dynamics is shown as a movement of the data points on the map.

2 Interactive Two-Level WEBSOM

In the original WEBSOM two-level architecture, the map of words was used to find synonyms and otherwise closely related words [1,13]. This procedure helps in lowering the dimensionality of the input vector for the document map and in finding relationships between semantically related documents even when different words are used to describe the same thing. The random projection method also proved to be feasible in dimensionality reduction [13].

Since the early developments, much more powerful computational resources and automatic term selection methods have become available. Moreover, there is an increasing number of Semantic Web based and other terminological resources. Therefore, in the Interactive Two-Level WEBSOM (I2WEBSOM), the terminology is extracted or defined beforehand. The map of words becomes effectively a map of terms in which each term is related to the domain(s) of the application at hand. The I2WEBSOM has a two-level architecture:

- The map of terms is calculated to enable interactive feature selection and weighting.
- The map of documents is calculated using terminology-based feature vectors where their weights can be changed using the first-level map. In this paper, we focus on the application of creating maps of people.

The I2WEBSOM prototype has been fully implemented in Javascript with a json interface to the person database (people.aalto.fi). The profiles of the people in the system consist currently of a subset of the staff in the six schools of Aalto University. The descriptive documents are a concatenation of each person's description, associated terms, and titles of their publications. The I2WEBSOM architecture and a snapshot of the functional system is shown in Fig. 1.

2.1 Maps of Words as a Feature Selection Tool

The WEBSOM method has also been used to support qualitative research [21]. It was concluded that the utility of the SOM in improving inference quality follows from the fact that the method can easily be used to generate multiple well-grounded perspectives on the data. These perspectives are not a collection of random views but form an organized whole [21]. In a map of documents, different points of view can be obtained through different weightings of the terms. A seemingly neutral starting point is to weight all features equally or to use a weighting scheme that is based only on statistical criteria such as different variants of tf-idf.

The user often has preferences which makes some conceptual domains covered by the text collection more relevant than the others. As one of the strengths of the maps of

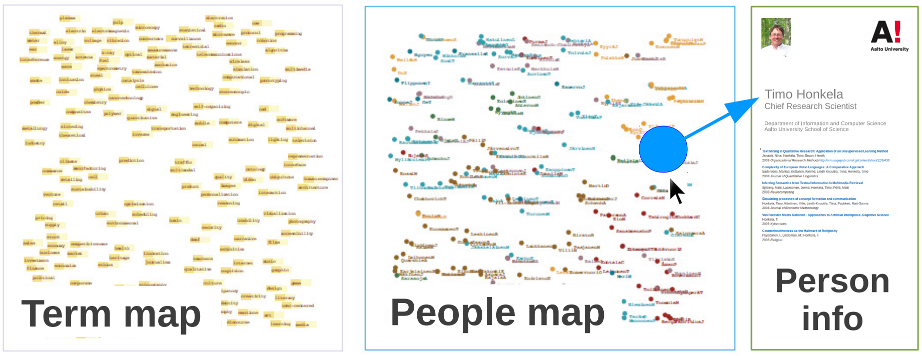


Fig. 1. An Interactive WEBSOM interface with three main elements: (1) A term map for choosing term weightings, (2) A dynamic people map, the order of which depends on the chosen term weights, and (3) Person information that can be viewed by clicking a person on the people map

documents is the possibility to obtain an overall view, this question of relevance is not related to individual queries but to the overall mapping function. Moreover, when the number of terms is in hundreds or thousands, one may wish to change the weights of groups of terms at once. A map of terms (Fig. 2) can be used to accomplish such task. Contextual information gives rise to a map in which related terms are close to other and groups of them form conceptual clusters. In an interactive interface, one can choose an area on the map and apply a weight changing operation on the selected terms. For instance, in relation to a map of university staff, one may choose to give increased weight to terms, e.g., related to business and software engineering. The effect of the change is such that the areas that contain persons working in these areas will be magnified.

2.2 Visualizing Map Organization

The vast majority of SOM-based data analysis and visualization tools are based on the idea that only the end result, an organized map is given to the user. The dynamic process of self-organization is typically shown only when the algorithm is demonstrated in educational settings. In I2WEBSOM, the organization and re-organization of the document map is shown to the user. The main motivation is to help the user in detecting the effects of the weight changes and to provide an understanding of the nature of dimensionality reduction.

The dynamic visualization of the organization process takes place by showing the position of each input vector x_j. The positions are changed based on the iterative updates of the model vectors $m_i(t)$. If the change of the best-matching units would be visualized straightforwardly, it would be impossible in practice to follow the process in sufficient detail. Therefore, the changes in the locations of the data points, loc_i, are updated based on the update rule $loc_i(t+1) = loc_i(t) + \beta[BMU_i(t)loc_i(t)]$. For the parameter β, the step size used in the visualization, the value 0.1 has been found to be appropriate. Due to the limitations of a printed document, the organization process is not shown here but examples can be found at $http://research.ics.aalto.fi/cog/websom/$.

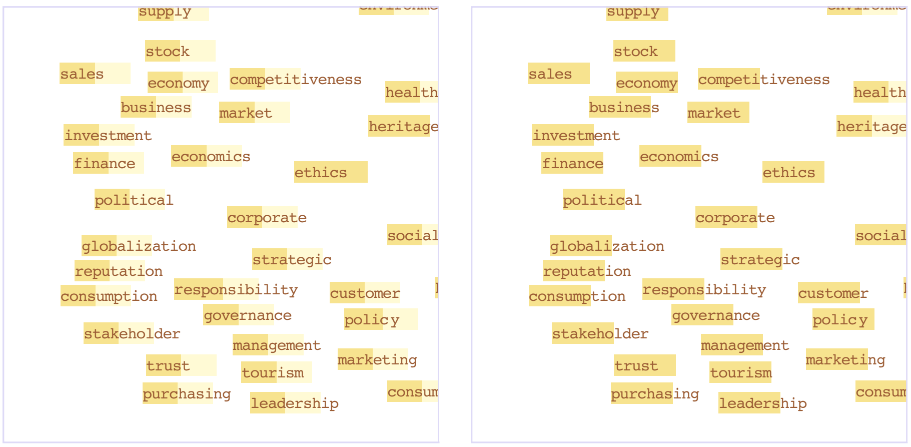


Fig. 2. An example of a map of terms where a part of the map is shown. Each term is associated with a slide bar that can be used to change the weight of the term. Also a group of terms can be manipulated simultaneously. On the left hand side, a neutral weighting is in use, and on the right hand side, the weights of a selected set of terms have been increased.

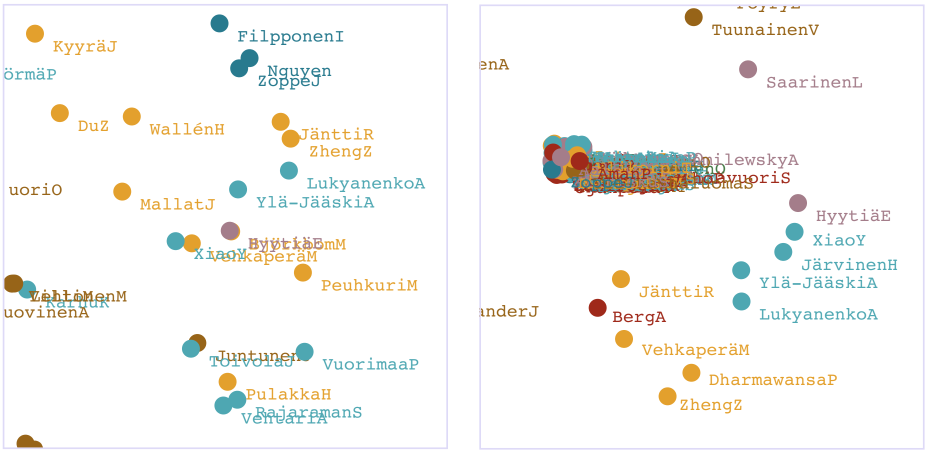


Fig. 3. Two versions of a zoomed map of people. The coloring of each node indicates the school of the person. On the left hand side, the map has been created with a neutral weighting in which each term has a similar weight. On the right hand side, the map has become restructured based on reweighting of the terms. One notable effect is that those people whose interests are defined by terms with a low weight are grouped tightly together. This helps in exploring the relevant parts of the map.

The end result of the organization process is in this case a map of people where two persons are close to each other if they have relatively similar interests or competences. Fig. 3 shows two examples of an area of a map of people.

3 Conclusions and Discussion

In this article, a novel development related to using the self-organizing map in the visualization of document collections has been presented. Different points of view into a document collection can be formed by the I2WEBSOM method. An interactive term map can be used in changing the weights of terms and terms groups.

The usefulness of the SOM in data visualization has been shown in practice through numerous applications as well as quantitatively [3]. There is also a large number of applications of the WEBSOM and closely related methods, and their usefulness has been shown in relation to information retrieval tasks [16]. The I2WEBSOM method increases the users' control over the organization of the document map and helps in creating transparency between data and the end result.

A quantitative evaluation of the I2WEBSOM method in a traditional manner would be extremely difficult as the motivation is to enable different points of view into the same data, based on interactions with the user. Moreover, the method has not been developed in one particular information retrieval task in mind but it can serve several purposes simultaneously. One may wish to obtain an overall view on the people in an organization or look for people with a particular interest or competence profile. Therefore, it remains a future task to determine what kind of user studies are needed to evaluate in a detailed manner the method presented in this paper. These are naturally beyond the scope of this paper. Instead, with this method we hope to indicate novel kinds of future application possibilities for unsupervised neural network and machine learning techniques.

Acknowledgments. We are grateful to Oliver Manner and Jan Fabritius who have developed the Aalto People web service and provided access to the data used in the experiments. The support from Philip Dean and Juhani Tenhunen is gratefully acknowledged as well as the collaboration with Jorma Laaksonen and Hannele Törrö in the earlier stages of the project.

References

1. Honkela, T., Kaski, S., Lagus, K., Kohonen, T.: Newsgroup exploration with WEBSOM method and browsing interface. Technical Report A32, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland (1996)
2. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (2001)
3. Venna, J., Kaski, S.: Local multidimensional scaling. Neural Networks 19(6), 889–899 (2006)
4. Ritter, H., Kohonen, T.: Self-organizing semantic maps. Biological Cybernetics (1989)
5. Harris, Z.: Distributional structure. Word 10(23), 146–162 (1954)
6. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. J. of Artificial Intelligence Research 37, 141–188 (2010)
7. Honkela, T., Pulkki, V., Kohonen, T.: Contextual relations of words in Grimm tales analyzed by self-organizing map. In: Proc. of ICANN 1995, Paris, EC2 et Cie, vol. 2, pp. 3–7 (1995)
8. Honkela, T., Vepsäläinen, A.M.: Interpreting imprecise expressions: Experiments with Kohonen's self-organizing maps and associative memory. In: Proc. of ICANN 1991, vol. 1, pp. 897–902 (1991)

9. Lin, X., Soergel, D., Marchionini, G.: A self-organizing semantic map for information retrieval. In: Proc. of the 14th ACM SIGIR, pp. 262–269 (1991)
10. Jäppinen, H., Honkela, T., Hyötyniemi, H., Lehtola, A.: A multilevel natural language processing model. Nordic Journal of Linguistics 11, 69–82 (1988)
11. Alkula, R., Honkela, T.: Development of text storage and information retrieval methods with natural language processing components. Final report of the FULLTEXT project (in Finnish). VTT, Espoo, Finland (1992)
12. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. J. of the American Society of Information Science 41, 391–407 (1990)
13. Kaski, S., Honkela, T., Lagus, K., Kohonen, T.: WEBSOM–self-organizing maps of document collections. Neurocomputing 21(1), 101–117 (1998)
14. Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V., Saarela, A.: Self organization of a massive document collection. IEEE Transactions on Neural Networks 11(3), 574–585 (2000)
15. Ong, T.H., Chen, H., Sung, W.K., Zhu, B.: Newsmap: a knowledge map for online news. Decision Support Systems 39(4), 583–597 (2005)
16. Saarikoski, J., Laurikkala, J., Järvelin, K., Juhola, M.: A study of the use of self-organising maps in information retrieval. Journal of Documentation 65(2), 304–322 (2009)
17. Ding, Y., Fu, X.: The research of text mining based on self-organizing maps. Procedia Engineering 29, 537–541 (2012)
18. Lagus, K.: Map of WSOM 1997 abstracts–alternative index. In: Proc. of WSOM 1997, vol. 97, pp. 4–6 (1997)
19. Honkela, T., Nordfors, R., Tuuli, R.: Document maps for competence management. In: Proc. of the Symposium on Professional Practice in AI, pp. 31–39 (2004)
20. Piazza, F., Strohmeier, S.: Domain-driven data mining in human resource management: A review. In: Proc. of ICDMW 2011, pp. 458–465 (2011)
21. Janasik, N., Honkela, T., Bruun, H.: Text mining in qualitative research application of an unsupervised learning method. Organizational Research Methods 12(3), 436–460 (2009)

Optimal Operation of Electric Power Production System without Transmission Losses Using Artificial Neural Networks Based on Augmented Lagrange Multiplier Method

George J. Tsekouras[1], Fotis D. Kanellos[2], Nikos E. Mastorakis[1,3], and Valeri Mladenov[4]

[1] Department of Electrical & Computer Science, Hellenic Naval Academy, Terma Hatzikyriakou, 18539 Piraeus, Greece
[2] Production Engineering & Management Department, Technical University of Crete, Technical University Campus, 73100 Chania, Greece
[3] English Language Faculty of Engineering, Technical University of Sofia, 8 St. Kl. Ohridski Blvd., 1000, Sofia Bulgaria
[4] Department of Theoretical Electrical Engineering, Technical University of Sofia, 8 St. Kl. Ohridski Blvd., 1000, Sofia Bulgaria
tsekouras_george_j@yahoo.gr, kanellos@mail.ntua.gr, mastor@wseas.org, valerim@tu-sofia.bg

Abstract. The optimal economic operation of a thermal electric power production system without considering transmission losses is a critical problem for ships, aircrafts, island power systems and it is usually solved with Lagrange method. In this paper, an alternative solution method is proposed using artificial neural networks (ANN) based on augmented Lagrange multiplier method with equality and inequality constraints. The respective theoretical analysis is presented, while a specific case study is studied. The advantages and disadvantages of the method are discussed and compared with the classical Lagrange method and ANN method based on external penalty functions.

Keywords: ANN, economic dispatch, thermal power system.

1 Introduction

Artificial neural networks (ANNs) have been applied to many power systems problems such as, short-term load forecasting [1], midterm energy forecasting [2], reliability analysis [3], dynamic security analysis [4], system protection [5], load shedding [6], power quality [7], transformer iron losses estimation [8], generators design optimization [9], estimation of insulators characteristics [10] etc. In previous work [11] the authors have developed an artificial neural network based on external penalty functions for the optimal operation of thermal electric power production system without transmission losses [12-13].

© Springer-Verlag Berlin Heidelberg 2013

The objective of this paper is to present an alternative method based on artificial neural networks specialized in the solution of continuous, nonlinear constrained optimization problems, exploiting augmented Lagrange multiplier method [14]. In section 2 the mathematical base of the optimal operation of a thermal electric power system is formed. In section 3 the mathematical base of the developed ANN based on augmented Lagrange multiplier method is presented while in section 4 indicative case studies are presented. Finally, in section 5 the advantages and the disadvantages of the classic Lagrange technique, the ANN method based on external penalty functions [11] and the presented method are commented.

2 Mathematical Base for Optimal Operation of All-Thermal Power System without Transmission Losses

The optimal operation of the thermal and of the hydro-thermal continental power systems with or without transmission losses, known as economic dispatch, has been already analyzed in the literature [12-13]. In this section the respective mathematical background of the optimal operation of an all-thermal power system without transmission system losses is presented briefly. It is assumed that the thermal system consists of N thermal generating units connected to the same bus. Let the j^{th} unit produces output active power $P_{\text{TH}j}$, bounded by the technically minimum active power, $\text{P}_{\text{minTH}j}$, and the technically maximum active power $\text{P}_{\text{maxTH}j}$ of the unit. The respective constraints are formulated as:

$$P_{\min \text{TH}j} \le P_{\text{TH}j} \le P_{\max \text{TH}j} \text{ for } \quad j = 1, 2, ..., N \tag{1}$$

The respective fuel cost of each unit is given by the function $F_{THj}(P_{THj})$, which is usually a polynomial of second or third order of $P_{\text{TH}j}$. This means that:

$$F_{\text{TH}j}\left(P_{\text{TH}j}\right) = a_j + b_j \cdot P_{\text{TH}j} + c_j \cdot P_{\text{TH}j}^2 + d_j \cdot P_{\text{TH}j}^3 \tag{2}$$

Where a_j, b_j, c_j, d_j are proper economic coefficients considered as known next.

For each time interval DT with constant load demand P_D the total fuel cost F_{tot} of the power system is the sum of the individual fuel costs of the units, and it is calculated by the following multivariable continuous differential equation:

$$F_{tot} = \sum_{j=1}^{N} F_{\text{TH}j}\left(P_{\text{TH}j}\right) \tag{3}$$

Active power balance without considering transmission losses can be written for each time interval DT, as:

$$\sum_{j=1}^{N} P_{\text{TH}j} = P_D \tag{4}$$

The target is to determine the generating levels of the units such that the total fuel cost $F_{tot,}$ is minimized and active power balance constraint is fulfilled. This means that the total fuel cost F_{tot} should be minimized taking into consideration the respective constraint of eq. (4). This problem can be solved by integrating eq. (4) in eq. (3) with the use of an unknown Lagrange multiplier, i.e. λ. Then eq. (3) is modified as:

$$L_{tot} = \sum_{j=1}^{N} F_{\mathrm{TH}j}\left(P_{\mathrm{TH}j}\right) - \lambda \cdot \left(\sum_{j=1}^{N} P_{\mathrm{TH}j} - P_D \right) \tag{5}$$

Solving Lagrange equation with classic Gauss-Seidel technique the determination of λ is achieved as it has been described in [11]. Afterwards the respective power production of the jth unit is calculated.

3 Mathematical Base for Continuous, Nonlinear, Constrained Optimization Process Using ANNs and Based on Augmented Lagrange Multiplier Technique

3.1 General

A constrained minimization problem can be stated as the following one: "Find $\vec{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$ which minimizes the continuous, non-linear scalar function $f(\vec{x}) = f(x_1, x_2, ..., x_n)$ subject to the equality constraints $h_i(\vec{x}) = 0$ for $i = 1, 2, ..., p$ and the inequality constraints $g_i(\vec{x}) \geq 0$ for $i = p+1, p+2, ..., m$". It is noted that inequality constraint $w(\vec{x}) \leq 0$ can be replaced by the inequality constraint $-w(\vec{x}) \geq 0$. The aforementioned problem can be transformed to an unconstrained problem having the following pseudo-cost energy function:

$$E(\vec{x}) = L(\vec{x}, \vec{\lambda}, \vec{k}) = \begin{cases} f(\vec{x}) + \sum_{i=1}^{p} \left[\lambda_i \cdot h_i(\vec{x}) + \frac{1}{2} \cdot k_i \cdot h_i^2(\vec{x}) \right] \\ + \sum_{i=p+1}^{m} \left[\lambda_i \cdot g_{\min\_i}(\vec{x}) + \frac{1}{2} \cdot k_i \cdot g_{\min\_i}^2(\vec{x}) - \frac{a}{2} \cdot \lambda_i^2 \right] \end{cases} \tag{6}$$

Where λ_i are the Lagrange multipliers (λ_i<0), k_i are the multiplication penalty factors (k_i>0), a is the regularization parameter (a≥0) for the case of an ill-conditioned problem and $g_{\min\_i}(\vec{x})$ are the modified inequality functions defined by:

$$g_{\min\_i}(\vec{x}) = \min\left\{ g_i(\vec{x}), -\frac{\lambda_i}{k_i} \right\} \text{ for } i = p+1, p+2, ..., m \tag{7}$$

If the first partial derivatives of the constraints $g_i(\vec{x})$ are continuous in $\mathbb{R}^n$ then this will also happen for the modified inequality functions $g_{\min\_i}(\vec{x})$ of eq. (10), as:

$$\frac{\partial g_{\min\_i}(\vec{x})}{\partial x_j} = S_i \cdot \frac{\partial g_i(\vec{x})}{\partial x_j} \tag{8}$$

Where S_i is a proper step function:

$$S_i = \begin{cases} 0, & if\ g_i(\vec{x}) \ge -\dfrac{\lambda_i}{k_i} \\ 1, & if\ g_i(\vec{x}) < -\dfrac{\lambda_i}{k_i} \end{cases} \quad \text{for } i = p+1, p+2, ..., m \tag{9}$$

The minimization of the energy function is achieved by solving the following system of differential equations:

$$\frac{dx_j}{dt} = -\mu_j \cdot \left(\begin{array}{l} \dfrac{\partial f}{\partial x_j} + \sum_{i=1}^{p} \left(\lambda_i + k_i \cdot h_i(\vec{x})\right) \cdot \dfrac{\partial h_i(\vec{x})}{\partial x_j} \\ + \sum_{i=p+1}^{m} S_i \cdot \left(\lambda_i + k_i \cdot g_i(\vec{x})\right) \cdot \dfrac{\partial g_i(\vec{x})}{\partial x_j} \end{array} \right) \text{ for } j = 1,2,...,n \,\&\, x_j(0) = x_j^{(0)} \tag{10}$$

$$\frac{d\lambda_i}{dt} = \rho_i \cdot h_i(\vec{x}) \text{ for } i = 1,2,..., p \,\&\, \lambda_i(0) = \lambda_i^{(0)} \tag{11}$$

$$\frac{d\lambda_i}{dt} = \rho_i \cdot \left(S_i \cdot g_i(\vec{x}) - \lambda_i \cdot \min\left\{ \frac{1-S_i}{k_i}, a \right\} \right) \text{ for } i = p+1,...,m \ \&\ \lambda_i(0) = \lambda_i^{(0)} \tag{12}$$

Where:

- t is the auxiliary parameter of time,
- μ_j is the ANN learning rate for the variable x_j upper limited by a constant positive number μ_0 defined by the following equation:

$$\mu_j(t) = \max\left\{ \mu_{init,j} \cdot \exp\left(-\frac{t}{T_{\mu 0,j}} \right), \mu_{0,j} \right\} \text{for } j = 1,2,...,n \,\&\, \mu_{init,j}, \mu_{0,j}, T_{\mu 0,j} > 0 \tag{13}$$

- k_i is the multiplication penalty factor lower limited by a constant positive number k_0 defined by the following equation:

$$k_i(t) = \min\left\{ \frac{k_{init,i}}{t}, k_{0,i} \right\} \text{for } i = 1,2,...,m \ \&\ k_{init,i}, k_{0,i} > 0 \tag{14}$$

- ρ_i is the ANN learning rate for the Lagrange variable λ_i lower limited by a constant positive number ρ_0 defined by the following equation:

$$\rho_i(t) = \min\left\{ \frac{\rho_{init,i}}{t}, \rho_{0,i} \right\} \text{for } i = 1,2,...,m \ \&\ \rho_{init,i}, \rho_{0,i} > 0 \tag{15}$$

3.2 Optimal Operation of All-Thermal Power System without Transmission Losses

The target is to minimize the total fuel cost F_{tot} of the power system estimated in eq. (3). In this minimization problem the unknown variable vector $\vec{x}$ consists of the power generation levels of the units $P_{\mathrm{TH}j}$, while the function $f\left(\vec{x}\right)$ represents total fuel cost function, F_{tot}:

$$\vec{x} = \left(P_{TH1}, P_{TH2}, \ldots, P_{TH,N}\right)^T \in \mathbb{R}^N \tag{16}$$

$$f\left(\vec{x}\right) = \sum_{j=1}^{N}\left(a_j + b_j \cdot P_{\mathrm{TH}j} + c_j \cdot P_{\mathrm{TH}j}^2 + d_j \cdot P_{\mathrm{TH}j}^3\right) \tag{17}$$

The unique active power balance equality without considering the losses of the transmission system defined in eq. (4) is:

$$h_1\left(\vec{x}\right) = \sum_{j=1}^{N} P_{\mathrm{TH}j} - P_D, \ p=1 \tag{18}$$

The technical constraints of the thermal units of eq. (1) are transformed to the following inequalities:

$$g_{1+i}\left(\vec{x}\right) = P_{\max \mathrm{TH}i} - P_{\mathrm{TH}i} \geq 0 \ \text{ for } i = 1, 2, \ldots, N \tag{19}$$

$$g_{1+N+i}\left(\vec{x}\right) = P_{\mathrm{TH}i} - P_{\min \mathrm{TH}i} \geq 0 \ \text{ for } i = 1, 2, \ldots, N \tag{20}$$

Hence, for N thermal units $2{\cdot}N$ inequality constraints are obtained.

The first partial derivatives of the minimization function and the constraints are the following:

$$\frac{\partial f}{\partial P_{\mathrm{TH}j}} = b_j + 2 \cdot c_j \cdot P_{THj} + 3 \cdot d_j \cdot P_{\mathrm{TH}j}^2 \ \text{ for } j = 1, 2, \ldots, N \tag{21}$$

$$\frac{\partial h_1}{\partial P_{\mathrm{TH}j}} = 1 \ \text{ for } j = 1, 2, \ldots, N \tag{22}$$

$$\frac{\partial g_i}{\partial P_{\mathrm{TH}j}} = \begin{cases} -1, & j = i-1 \\ 0, & j \neq i-1 \end{cases} \ \text{ for } i = 2, 3, \ldots, N+1, \ j = 1, 2, \ldots, N \tag{23}$$

$$\frac{\partial g_i}{\partial P_{\mathrm{TH}j}} = \begin{cases} 1, & j = i-N-1 \\ 0, & j \neq i-N-1 \end{cases} \ \text{ for } i = N+2, N+3, \ldots, 2 \cdot N+1, \ j = 1, 2, \ldots, N \tag{24}$$

The ANN technique used in this study is an iterative process, where eq. (10)-(12) can be easily transformed into the following iterative, discrete-time equations:

$$P_{\mathrm{TH}j}^{(\ell+1)} = \left\{ \begin{array}{l} P_{\mathrm{TH}j}^{(\ell)} - \mu_j(\ell)\cdot\dfrac{\partial f\left(\vec{x}^{(\ell)}\right)}{\partial P_{\mathrm{TH}j}} - \mu_j(\ell)\cdot\left(\lambda_1(\ell)+k_1(\ell)\cdot h_1(\vec{x})\right)\cdot\dfrac{\partial h_1(\vec{x})}{\partial P_{\mathrm{TH}j}} \\ -\mu_j(\ell)\cdot\sum\limits_{i=2}^{2\cdot N+1} S_i(\ell)\cdot\left(\lambda_i(\ell)+k_i(\ell)\cdot g_i\left(\vec{x}^{(\ell)}\right)\right)\cdot\dfrac{\partial g_i\left(\vec{x}^{(\ell)}\right)}{\partial P_{\mathrm{TH}j}} \end{array} \right\} \quad (25)$$

$$\text{for } j = 1,2,\ldots,N$$

$$\lambda_1(\ell+1) = \lambda_1(\ell) + \rho_1(\ell)\cdot h_1(\vec{x}(\ell)) \quad (26)$$

$$\lambda_i(\ell+1) = \lambda_i(\ell) + \rho_i(\ell)\cdot\left(S_i(\ell)\cdot g_i(\vec{x}(\ell)) - \lambda_i(\ell)\cdot\min\left\{\frac{1-S_i(\ell)}{k_i(\ell)}, a\right\}\right) \quad (27)$$

$$\text{for } i = 2,3\ldots,2\cdot N+1$$

Where, ℓ is the discrete variable for time or alternatively the “training epochs of ANN technique”. The proposed ANN training process is terminated if one of the following two stopping criterions is reached:

- the power generation levels of the units $P_{\mathrm{TH}j}$ are stabilized (the variation between two epochs should be smaller than convergence limit, "limit_convergence"),
- the maximum number of epochs is exceeded (larger than "max_*epochs*").

4 Application of the Proposed Method

A simple thermal electric power system without transmission losses is assumed. It consists of three thermal units with the following fuel cost functions and technical constraints:

$$F_{\mathrm{TH},1}\left(P_{\mathrm{TH},1}\right) = 2700 + 100\cdot P_{\mathrm{TH},1} + 0{,}03\cdot P_{\mathrm{TH},1}^2 + 10^{-4}\cdot P_{\mathrm{TH},1}^3 \ [\text{m.u./h}] \quad (28)$$

$$F_{\mathrm{TH},2}\left(P_{\mathrm{TH},2}\right) = 8760 + 75\cdot P_{\mathrm{TH},2} + 0{,}1\cdot P_{\mathrm{TH},2}^2 + 5\cdot 10^{-5}\cdot P_{\mathrm{TH},2}^3 \ [\text{m.u./h}] \quad (29)$$

$$F_{\mathrm{TH},3}\left(P_{\mathrm{TH},3}\right) = 4800 + 90\cdot P_{\mathrm{TH},3} + 0{,}03\cdot P_{\mathrm{TH},3}^2 + 8\cdot 10^{-5}\cdot P_{\mathrm{TH},3}^3 \ [\text{m.u./h}] \quad (30)$$

Where $50 \le P_{\mathrm{TH},1}, P_{\mathrm{TH},2} \le 350\ [\text{MW}]$, $50 \le P_{\mathrm{TH},3} \le 450\ [\text{MW}]$ and “m.u.” is the monetary unit. Two load levels are examined: 460 MW and 260 MW. The last one activates the technical constraints of the thermal units. For ANN training the following parameters values have been used, which have been chosen after a short optimization process for each parameter separately:

$$\mu_{init,j} = 10^{-3} \quad \mu_{0,j} = 1.000 \quad T_{\mu 0,j} = 200 \ \text{for } j\text{=1, 2, 3}$$

$$k_{init,i} = 0.001 \quad k_{0,i} = 1.000 \quad \rho_{init,i} = 0.001 \quad \rho_{0,i} = 1.000 \ \text{for } i\text{=1, …, 7}$$

$$\text{limit\_convergence} = 10^{-5} \quad \text{max\_}epochs = 10000$$

Power generation levels of the units are initialized as following:

$$P_{\mathrm{TH}j} = 0.5 \cdot \left(P_{\min \mathrm{TH}j} + P_{\max \mathrm{TH}j} \right) \text{ for } j = 1,2,3 \tag{31}$$

In case of 460 MW demand load the generating levels of the units are obtained after 330 epochs, approximately. In Figures 1 and 2 the generating levels of the thermal units and the total fuel cost with respect to the training epochs are presented. In case of 260 MW load demand the generating levels of the units are obtained after 1310 epochs, approximately. In both cases the active power balance error is smaller than 0.01 MW.

In Table 1 the results of the generation levels of the thermal units, the power balance error (difference between produced power by the units and load demand), the epochs and the total operation cost of the system are registered for both of the examined cases (460 MW and 260 MW of load demand) and for three minimization techniques (classical Lagrange optimization method, ANN based on external penalty functions [11], proposed ANN based on augmented Lagrange method).

It is concluded that:

- The three methods give practically the same results.
- Both ANN methods need significant more epochs for convergence than the classical technique.
- The examined method exploiting ANN based on augmented Lagrange method satisfies exactly the active power balance constraint.
- The ANN based on external penalty functions seems to be slightly superior to the classical technique in terms of fuel cost minimization, which, however, is not true. This happens because in both load levels the total power production of the units is slightly smaller than total load demand.

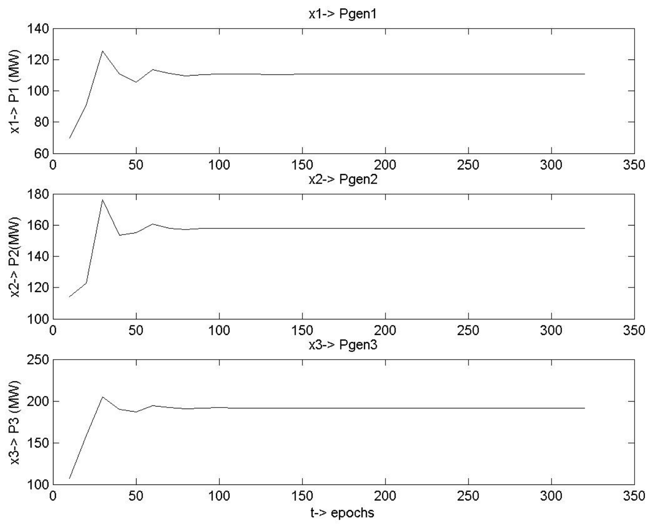


Fig. 1. Case of demand load 460 MW: the generating levels of the thermal units with respect to the number of epochs

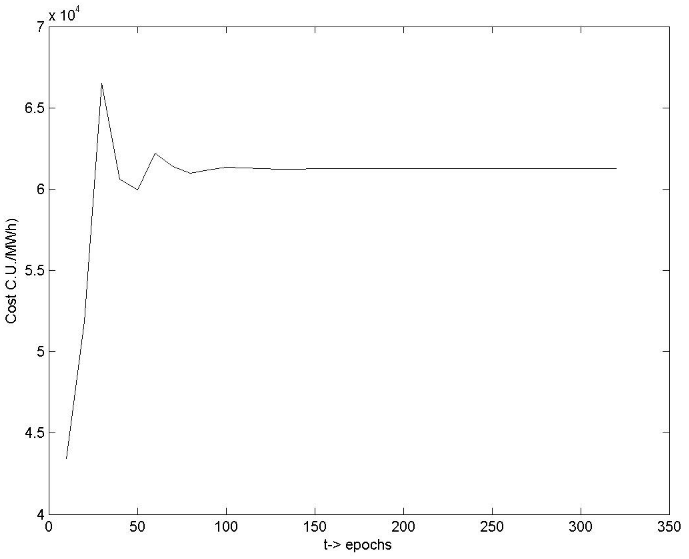


Fig. 2. Case of demand load 460 MW: the total fuel cost with respect to the number of epochs

Table 1. Comparison of classical Lagrange optimization method, ANN based on external penalty functions [11], proposed ANN based on augmented Lagrange method

Method	Classic	ANN [11]	Proposed ANN	Classic	ANN [11]	Proposed ANN
Target Load		460 MW			260 MW	
P_{TH1}[MW]	110.58	110.97	110.58	50.00	49.99	49.99
P_{TH2}[MW]	157.83	157.50	157.83	108.78	109.27	108.80
P_{TH3}[MW]	191.58	190.80	191.58	101.22	100.52	101.21
Error [MW]	0.01	-0.73	0.01	0.00	-0.22	0.00
Epochs - iterations [-]	8	12647	329	13	824	1310
Cost[m.u./h]	61252	61171	61252	40254	40232	40254

5 Conclusions

This paper presents an ANN based on augmented Lagrange method applied to a thermal electric power system without considering transmission losses. The respective results are satisfactory if the proper parameters of learning rate and penalty multiplicative factors are properly selected. The proposed method is characterized by one drawback as the number of iterations (epochs) needed is significant larger than those required in classical Lagrange technique, while both methods are equivalent for the rest results. However, the behavior of the proposed ANN is significantly better than ANN based on external penalty functions [11], as active power balance constraint is satisfied completely and the system marginal cost can be calculated directly by the Lagrange multiplier λ_1 of the respective equality constraint.

References

1. Tsekouras, G.J., Kanellos, F.D., Kontargyri, V.T., Tsirekis, C.D., Karanasiou, I.S., Elias, C.N., Salis, A.D., Mastorakis, N.E.: A comparison of Artificial Neural Networks algorithms for short term load forecasting in Greek intercontinental power system. In: WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing (CSECS 2008), Puerto De La Cruz, Canary Islands, Spain (2008)
2. Tsekouras, G.J., Hatziargyriou, N.D., Dialynas, E.N.: An Optimized Adaptive Neural Network for Annual Midterm Energy Forecasting. IEEE Transactions on Power Systems 21, 385–391 (2006)
3. Amzady, N., Ehsan, M.: Evaluation of power systems reliability by an artificial neural network. IEEE Transactions on Power Systems 14, 287–292 (1999)
4. Sobajic, D., Pao, Y.: Artificial Neural-Net based Dynamic Security Assessment for Electric Power Systems. IEEE Tans. on Power Systems 4, 220–227 (1989)
5. Bachmann, B., Novosel, D., Hart, D., Hu, Y., Saha, M.M.: Application of Artificial Neural Networks for Series Compensated Line Protection. In: Proc. of ISAP 1996, Orlando, Florida (1996)
6. Novosel, D., King, R.L.: Using an Artificial Neural Network for Load Shedding to alleviate overloaded lines. IEEE Tans. on Power Delivery 9, 425–433 (1994)
7. Mori, H., Itou, K., Uematsu, H., Tsuzuki, S.: An Artificial Neural Net Based Method for Predicting Power System Voltage Harmonics. IEEE Tans. on Power Delivery 7, 402–409 (1992)
8. Georgilakis, P.S., Hatziargyriou, N.D., Doulamis, N.D., Doulamis, A.D., Kollias, S.D.: Prediction of Iron Losses of Wound Core Distribution Transformers Based on Artificial Neural Networks. Neurocomputing 23, 15–29 (1998)
9. Tsekouras, G., Kiartzis, S., Kladas, A.G., Tegopoulos, J.A.: Neural Network Approach Compared to Sensitivity Analysis Based on Finite Element Technique for Optimization of Permanent Magnet Generators. IEEE Transactions on Magnetics 37, 3618–3621 (2001)
10. Kontargyri, V.T., Gialketsi, A.A., Tsekouras, G.J., Gonos, I.F., Stathopoulos, I.A.: Design of an Artificial Neural Network for the Estimation of the Flashover Voltage on Insulators. Electrical Power Systems Research 77, 1532–1540 (2007)
11. Tsekouras, G.J., Kanellos, F.D., Tsirekis, C.D., Mastorakis, N.E.: Optimal Operation of Thermal Electric Power Production System without Transmission Losses: An Alternative Solution using Artificial Neural Networks based on External Penalty Functions. In: 12th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Databases (AIKED 2013), Cambridge, England (2013)
12. Momoh, J.A.: Electric Power System Applications of Optimization. Marcen Dekker Inc., New York (2001)
13. Saccomanno, F.: Electric Power Systems, Analysis and Control. Wiley Interscience, IEEE Press, Piscataway, USA (2003)
14. Cichocki, A., Unbehauen, R.: Neural Networks for Optimization and Signal Processing. John Wiley & Sons, Stuttgart (1993)

An Echo State Network with Working Memories for Probabilistic Language Modeling

Yukinori Homma and Masafumi Hagiwara

The Department of Information and Computer Science, Keio University,
Hiyoshi 3-14-1, Kohoku-ku, Yokohama, 223-8522 Japan
{homma,hagiwara}@soft.ics.keio.ac.jp

Abstract. In this paper, we propose an ESN having multiple timescale layer and working memories as a probabilistic language model. The reservoir of the proposed model is composed of three neuron groups each with an associated time constant, which enables the model to learn the hierarchical structure of language. We add working memories to enhance the effect of multiple timescale layers. As shown by the experiments, the proposed model can be trained efficiently and accurately to predict the next word from given words. In addition, we found that use of working memories is especially effective in learning grammatical structure.

Keywords: Probabilistic language model, ESNs, working memory.

1 Introduction

Probabilistic language models are often used in natural language processing such as machine translation, speech recognition and orthographic error correction. These models are based on the Markov assumption, which means that they model the conditional probability distribution of the next word from a given sequence of words. For decades N-gram models are one of the most popular language modeling approach due to their simplicity and good modeling performance[1]. However, they are criticized for they capture only superficial linguistic structure and have no notion of grammar at distances greater than N[2]. Moreover, N-gram language modeling has the problems of data sparseness. The N-gram matrix for any given training corpus is bound to have a very large number of cases of putative zero probability N-grams[3].

Recently, several neural language models have been proposed[4][5][6]. Recurrent Neural Networks(RNN) are used as the basis of these models since Elman[7] showed their representational power to learn language. They can take into account arbitrary long dependency between words and represent the context of a sentence in their recurrent layer compactly. Therefore, RNN has ability to outperform N-gram models. In fact, MTRNN[5] model with a multiple timescale layer has shown to have superior performance in the language model tasks.

Echo State Networks(ESNs) proposed by Jaeger[8] are a type of three-layered RNN. They have the recurrent hidden layer whose connections are sparse, random and fixed. Only connections leading to the output layer could be changed

© Springer-Verlag Berlin Heidelberg 2013

during training. The main advantage of these approaches is the possibility of exploiting the processing capability of an RNN structure and, at the same time, of avoiding the difficulties associated with the RNN training process[9]. ESN models outperformed classical fully trained RNNs in many tasks[10].

In this paper, we propose a novel ESN having multiple timescale layer and working memories(WMs). Working memories provide mechanisms for storing, maintaining, retrieving and deleting information[11]. They maximize the effect of multiple timescale layers. We show that the working memories are effective in fast and accurate learning. In addition, we found that use of several working memories effectively contribute to acquiring the hierarchical structure in input sentences.

This paper is organized as follows: Section 2 gives the proposed model based on an ESN with working memories. Section 3 explains the experiments and the results. Finally we conclude with the discussion on the results in section 4.

2 Model

In this section, we present an ESN model with multiple timescale layer and working memories. ESNs are recent development in the field of recurrent nerural networks. They are composed of three layers: an input layer $\boldsymbol{u} \in \mathbb{R}^K$, a recurrent layer $\boldsymbol{x} \in \mathbb{R}^N$ called reservoir, and an output layer $\boldsymbol{y} \in \mathbb{R}^L$.

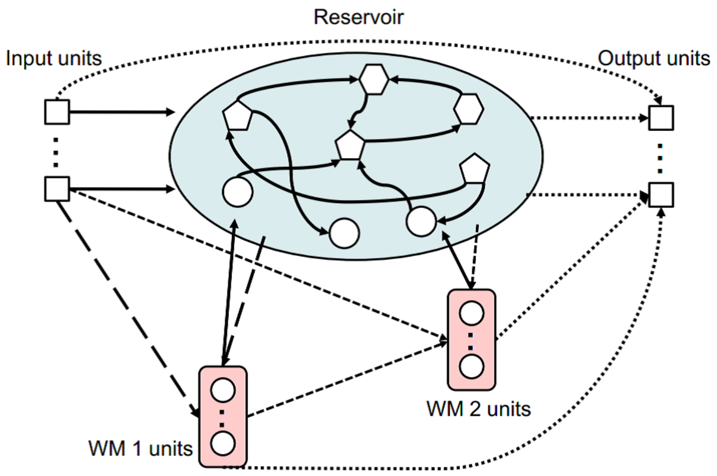


Fig. 1. The proposed model. The reservoir is composed of three neuron groups. The IO, Cf and Cs neuron groups are shown each as round, pentagon and hexagon shaped nodes. Connections with solid line are fixed while the others are trained.

The proposed model is shown in Fig. 1. The reservoir of the proposed model is composed of three neuron groups, each with an associated time constant τ.

The three neuron groups consist of the input-output(IO)$\boldsymbol{x}_{io} \in \mathbb{R}^{N_{io}}$, Fast Context(Cf)$\boldsymbol{x}_{cf} \in \mathbb{R}^{N_{cf}}$, and Slow Context(Cs)$\boldsymbol{x}_{cs} \in \mathbb{R}^{N_{cs}}$($N_{io}+N_{io}+N_{io} = N$). We set $\tau = 2, 5, 70$ for each IO, Cf, Cs neuron group in our experiments. These values are the same as [12]. The number of IO, Cf, Cs neuron units is each 300, 400, 110. These values are just ten times as large as that of [12].

A working memory is obtained by a set of special output units. Please refer to [11] for more information. We employed two working memories($\boldsymbol{m}_1 \in \mathbb{R}^{M_1}$, $\boldsymbol{m}_2 \in \mathbb{R}^{M_2}$) to the ESN model with multiple timescale layer. We make the one of them($\boldsymbol{m}_1$) store information of sentences and the other($\boldsymbol{m}_2$) store that of words.

$\boldsymbol{W}^{in} \in \mathbb{R}^{N \times K}$ and $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ are the weights matrices for the input connections and internal connections. $\boldsymbol{W}^{b1} \in \mathbb{R}^{N \times M_1}$ and $\boldsymbol{W}^{b2} \in \mathbb{R}^{N \times M_2}$ are the matrices collecting the feedback weights from each of working memory units to the reservoir. These matrices are fixed during training.

$\boldsymbol{W}^{in}$, $\boldsymbol{W}^{b1}$ and $\boldsymbol{W}^{b2}$ weights are non-zero with a probability of 0.4 and randomly chosen to be either 0.5 or -0.5 with equal probability. $\boldsymbol{W}$ are non-zero with a probability of 0.4 and uniformly sampled from [-1, 1]. The connections between IO neuron and Cs neuron groups are set to zero. $\boldsymbol{W}$ are scaled to have a spectral radius of 0.98.

$\boldsymbol{W}^{m_1} \in \mathbb{R}^{M_1 \times (N+K+M_1)}$, $\boldsymbol{W}^{m_2} \in \mathbb{R}^{M_2 \times (N+K+M_1+M_2)}$ and $\boldsymbol{W}^{out} \in \mathbb{R}^{L \times (N+K+M_1+M_2)}$ are the weights matrices for the 1st memory connections, 2nd memory connections and output connections, respectively. These three matrices can be trained in the learning steps.

The system update equations are the following.

$$\boldsymbol{z}(t) = \boldsymbol{W}^{in}\boldsymbol{u}(t) + \boldsymbol{W}\boldsymbol{x}(t-1) + \boldsymbol{W}^{b1}\boldsymbol{m_1}(t-1) + \boldsymbol{W}^{b2}\boldsymbol{m_2}(t-1). \tag{1}$$

$$x_i(t) = \begin{cases} 0. & (t = 0 \wedge i \notin I_{Csc}) \\ Csc_i. & (t = 0 \wedge i \in I_{Csc}) \\ (1 - \frac{1}{\tau_i})x_i(t-1) + \frac{1}{\tau_i}f(z_i(t)). & (otherwise) \end{cases} \tag{2}$$

$$\boldsymbol{m_1}(t) = f^{out}(\boldsymbol{W}^{\boldsymbol{m_1}}(\boldsymbol{u}(t), \boldsymbol{x}(t), \boldsymbol{m_1}(t-1))). \tag{3}$$

$$\boldsymbol{m_2}(t) = f^{out}(\boldsymbol{W}^{\boldsymbol{m_2}}(\boldsymbol{u}(t), \boldsymbol{x}(t), \boldsymbol{m_1}(t), \boldsymbol{m_2}(t-1))). \tag{4}$$

$$\boldsymbol{y}(t) = f^{out}(\boldsymbol{W}^{\boldsymbol{out}}(\boldsymbol{u}(t), \boldsymbol{x}(t), \boldsymbol{m_1}(t), \boldsymbol{m_2}(t))). \tag{5}$$

- $\boldsymbol{z}$: internal state of reservoir($\boldsymbol{z} \in \mathbb{R}^N$)
- $z_i(t)$: internal state of i th neuron at step t
- $x_i(t)$: value of i th reservoir neuron at step t
- I_{Csc} : neuron index set of Csc
- Csc_i : initial state of i th Csc neuron
- τ_i : time constant of i th neuron

Here, f and f^{out} give the activation functions of the internal units. We use the hyperbolic tangent as f and the sigmoid function as f^{out}. We choose 60 neurons from Cs neurons to be used as the Controlling Slow Context(Csc), whose initial states determine the network's behavior. The Csc value is independently prepared for each training sequence. We can recognize each of target sentences because the value represents the target sequence. Please refer to [5] for the detail.

The memory connections($\boldsymbol{W}^{m_1}, \boldsymbol{W}^{m_2}$), output connection($\boldsymbol{W}^{out}$) and initial states($\boldsymbol{Csc}$) are updated as follows.

$$\boldsymbol{v}(t) = \boldsymbol{W}^{in}\boldsymbol{u}(t) + \boldsymbol{W}\boldsymbol{x}(t-1) + \boldsymbol{W}^{b1}\boldsymbol{m_1}(t-1). \tag{6}$$

$$w_{ij}^{m_1}(t+1) = w_{ij}^{m_1}(t) + \eta(m_{1i}^{target}(t) - m_{1i}(t))f'^{out}(v_i(t))x_j(t) - \lambda w_{ij}^{m_1}(t). \tag{7}$$

$$w_{ij}^{m_2}(t+1) = w_{ij}^{m_2}(t) + \eta(m_{2i}^{target}(t) - m_{2i}(t))f'^{out}(z_i(t))x_j(t) - \lambda w_{ij}^{m_2}(t). \tag{8}$$

$$w_{ij}^{out}(t+1) = w_{ij}^{out}(t) + \eta(y_i^{target}(t) - y_i(t))f'^{out}(z_i(t))x_j(t) - \lambda w_{ij}^{out}(t). \tag{9}$$

$$Csc_i^{new} = Csc_i^{old} + \eta f'(x_i(t))\sum_{j\in\boldsymbol{y}} w_{ji}(y_i^{target}(t) - y_i(t)). \tag{10}$$

- $\boldsymbol{v}$: internal state of 1st memory units.
- w_{ij} : connection weight from j th to i th neuron
- η : learn constant number
- $m_i^{target}(t)$: value of i th memory neuron at step t for target
- $m_i(t)$: value of i th memory neuron at step t
- $y_i^{target}(t)$: value of i th neuron at step t for target sentence
- $y_i(t)$: value of i th output neuron at step t
- λ : constant number for regularization

We introduce the penalty term for regularization in the learning. The regularization term is used to penalize large wights. We can reduce the influence of the over-fitting by adding this term.

The learning of proposed model is divided into 3 steps. At first, only $\boldsymbol{W}^{m_1}$ that is illustrated as longer dashed line in Fig. 1 is computed to estimate the type of grammar. And then $\boldsymbol{W}^{m_2}$ as shorter dashed line in Fig. 1 is calculated to estimate the part of speech. After learning of working memories, we compute $\boldsymbol{W}^{out}$ as dotted line in Fig. 1 to predict the next word as previously explained. In all of these cases, the input sequences are generated in the same way.

Table 1. Lexicon

Category	Nonterminal symbol	Words
Verb (intransitive)	V_I	jump, run, walk
Verb (transitive)	V_T	kick, punch, touch
Noun	N	ball, box
Article	ART	a, the
Adverb	ADV	quickly, slowly
Adjective (size)	ADJ_S	big, small
Adjective (color)	ADJ_C	blue, red, yellow

Table 2. Regular grammar

Sentence	Noun phrase	Adverb phrase
S → V_I	NP → ART N	ADJ → ADJ_S
S → V_I ADV	NP → ART ADJ N	ADJ → ADJ_C
S → V_T NP		ADJ → ADJ_S ADJ_C
S → V_T NP ADV		

3 Experiments

3.1 Target Language

In this experiment, we used the same language set used in [5]. The language set contains 17 words in 7 categories(Table 1) and a regular grammar consisting of 9 rules(Table 2). We used a set of characters($\boldsymbol{C}$) composed of the 26 letters in the alphabet('a' to 'z') and two other symbols(space, period). We defined each character($c_i \in \boldsymbol{C}$) as a character corresponding to the ith neuron($u_i \in I_{IO}$). The i th neuron has value 1 when the corresponding word is presented, 0.1 otherwise. We made m_1 and m_2 memories learn the type of grammar and the part of speech. We defined the type of grammar as 4 sentence rules in Table 2. We also defined the type of part of speech as 7 categories in Table 1. These memory neurons are encoded in the same way as input neurons. The model decides the next word from the result of an arg-max function on the output layer.

In the experiments, we set the two constants as follows: $\eta = 0.01$, $\lambda = exp(-18)$. We generated 100 different sentences from the regular grammar. We used 80 sentences to train the proposed model. The number of learning epochs is 2,000 for each memory connection, 10,000 for the output connection. We tested the trained model's capability using all of the 100 sentences. The testing procedure was as follows.

(i) Recognition: Calculate $\boldsymbol{Csc}$ using a sentence.
(ii) Generation: Generate a sentence from the $\boldsymbol{Csc}$ gained in (i).
(iii) Comparison: Compare the original and generated sentence.

In the recognition step, we calculate $\boldsymbol{Csc}$ using a given sentence. We set $\boldsymbol{Csc}$ as the one of the sentence that was the most similar to the given sentence in 80 training sentences.

3.2 Results

Table 3 shows the best result of sentence emulation task.

Table 3. Results of sentence emulation task

Model type	Correct next word	Correct sentences
Proposed model without WMs	0.98 (1891/1924)	72/100
Proposed model	0.99 (1910/1924)	88/100

As can be seen from Table 3, working memories contribute to obtain better performance of the sentence emulation task. The percentage of calculating next word is relatively high because there are only a few misspelled words in a sentence. Although we used 80 sentences to train the model without working memories, it could correctly emulate only 72 sentences. Table 3 implies that the learning process in the model without working memories is insufficient.

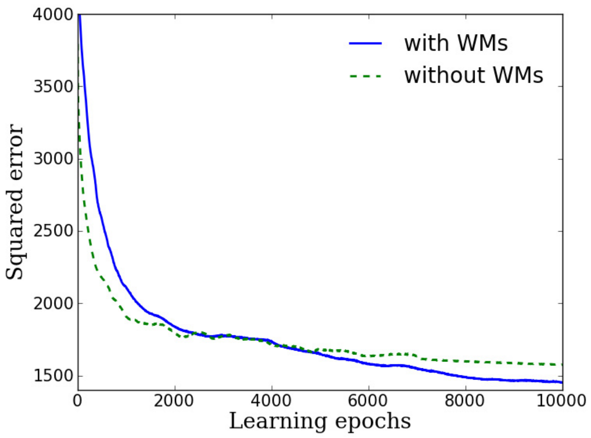


Fig. 2. Squared errors between $\boldsymbol{y}$ and $\boldsymbol{y}^{target}$ for the proposed model and the model without working memories. The horizontal line shows the number of learning epochs.

Squared errors between $\boldsymbol{y}$ and $\boldsymbol{y}^{target}$ for each model are illustrated in Fig. 2. The squared error of model with working memories is larger than the model without working memories in the beginning of learning because connections to

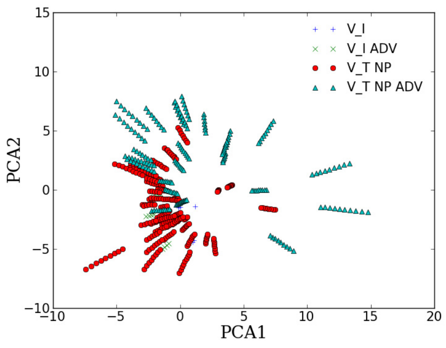


Fig. 3. Without WMs

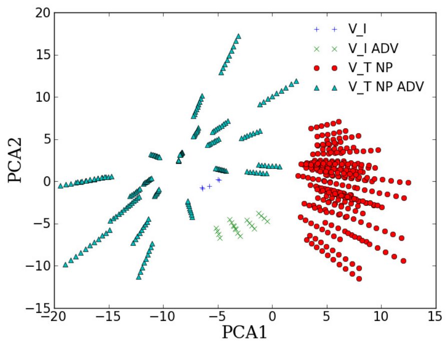


Fig. 4. With WMs

working memories have been already trained. Nevertheless, the value decreases quickly with progression of learning compared to the other one. We can also see that the squared error of the network with memories decreases to lower values compared to the network without memories. Therefore working memories are effective in learning to calculate the next word.

We analyzed the neural activation patterns when the proposed model generated sentences. We used the principle component analysis(PCA) to understand the neural patterns. In Fig. 3 and 4, we can see transitions of Cs neuron group activation for all of the words in 100 sentences.

Fig. 3 shows the neuron activation of the model without working memories. Each segment looks like a line represents one of sentences. We can see that the transitions of sentences are scattered randomly. There is little difference in the types of grammar.

Fig. 4 shows the neuron activation of the proposed model. Compared to Fig. 3, sentences are clearly clustered based on their categories. The sentences in the same category are represented in a similar way. We claim that the use of working memories has heavy influence on the transitions of neuron activation. Working memories work the reservoir to learn grammatical structures in input sentences.

4 Conclusion

In this paper, we proposed a probabilistic language model based on an ESN with working memories. The proposed model has a reservoir composed of three layers. Working memories have an effect to enhance the reservoir ability to store information of the sentence structure. As the result of the experiments, it is shown that the proposed model can be trained efficiently and accurately to predict the next word from given words.

We also analyzed the neural activation patterns of the proposed model using PCA. The results show that the sentences are clustered based on their grammatical structure. We conclude that use of working memories on ESN model is effective in learning grammatical structure.

For our future work, we intend to apply the proposed model to the task to learn larger grammatical structure. ESNs have potential to deal with large data due to their learning ability. We expect the proposed model can generate more natural sentences by learning from documents on the Web.

References

1. Arisoy, E., Sainath, T.N., Kingsbury, B., Ramabhadran, B.: Deep neural network language models. In: Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, WLM 2012, Stroudsburg, PA, USA, pp. 20–28. Association for Computational Linguistics (2012)
2. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education (2003)
3. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 1st edn. Prentice Hall PTR, Upper Saddle River (2000)
4. Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., Khudanpur, S.: Extensions of recurrent neural network language model. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5528–5531 (2011)
5. Hinoshita, W., Arie, H., Tani, J., Okuno, H.G., Ogata, T.: Emergence of hierarchical structure mirroring linguistic composition in a recurrent neural network. Neural Networks 24(4), 311–320 (2011)
6. Tong, M.H., Bickett, A.D., Christiansen, E.M., Cottrell, G.W.: Learning grammatical structure with Echo State Networks. Neural Networks 20(3), 424–432 (2007)
7. Elman, J.L.: Learning and development in neural networks: The importance of starting small. Cognition 48(1), 71–99 (1993)
8. Jaeger, H.: The" echo state" approach to analysing and training recurrent neural networks-with an erratum note, vol. 148. German National Research Center for Information Technology GMD Technical Report, Bonn (2001)
9. Boccato, L., Lopes, A., Attux, R., Zuben, F.J.V.: An extended echo state network using Volterra filtering and principal component analysis. Neural Networks 32, 292–302 (2012)
10. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review 3(3), 127–149 (2009)
11. Pascanu, R., Jaeger, H.: A neurodynamical model for working memory. Neural Networks 24(2), 199–207 (2011)
12. Yamashita, Y., Tani, J.: Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: A Humanoid Robot Experiment. PLoS Comput. Biol. 4(11) (2008)

Using the Analytic Feature Framework for the Detection of Occluded Objects

Marvin Struwe[1], Stephan Hasler[2], and Ute Bauer-Wersing[1]

[1] University of Applied Sciences Frankfurt am Main, Germany
[2] Honda Research Institute Europe GmbH, Offenbach, Germany
{mstruwe,ubauer}@fb2.fh-frankfurt.de,
stephan.hasler@honda-ri.de

Abstract. In this paper we apply the analytic feature framework, which was originally proposed for the large scale identification of segmented objects, for object detection in complex traffic scenes. We describe the necessary adaptations and show the competitiveness of the framework on different real-world data sets. Similar to the current state-of-the-art, the evaluation reveals a strong degradation of performance with increasing occlusion of the objects. We shortly discuss possible steps to tackle this problem and numerically analyze typical occlusion cases for a car detection task. Motivated by the fact that most cars are occluded by other cars, we present first promising results for a framework that uses separate classifiers for unoccluded and occluded cars and takes their mutual response characteristic into account. This training procedure can be applied to many other trainable detection approaches.

Keywords: Object detection, Supervised learning, Occlusion handling.

1 Introduction

Despite extensive efforts the visual detection of objects in natural scenes is still not robustly solved. The current best approaches usually extract unspecific local features and apply a powerful classifier directly on top. So e.g. the combination of Histograms of Oriented Gradients (HOG) [2] with a Support Vector Machine (SVM) is reported to yield good performance in various detection benchmarks. In contrast to this are methods that put effort in learning a more problem-specific feature representation, on top of which a very simple classifier can be used for discrimination. An example for such a method is the analytic feature architecture proposed in [5], which showed high performance for large-scale identification of segmented object views. In this paper we show that such an architecture can also provide competitive results in detection tasks.

One main problem for systems acting in real world is that objects are often occluded. This affects currently used object representations in different ways. E.g. the methods that aggregate local features in a voting manner like [6,7] are usually trained with unoccluded views. During recognition they can deal with arbitrary occlusion patterns as long as sufficiently many features can still be

© Springer-Verlag Berlin Heidelberg 2013

detected. In contrast to this are the methods that train holistic object templates in a discriminative manner like [2,5]. When training on unoccluded and testing on occluded views these approaches show a much stronger relative decrease in performance. The reason for this is the stronger specialization on the training problem by focusing resources on differences between classes. However, in general the voting methods perform worse than the discriminative ones, whenever test and training set do not show such systematic differences, as discussed in [11,3].

To exploit the benefits of discriminative approaches also for occluded objects, one could simply train them with occluded and unoccluded views. But this will likely reduce the performance for unoccluded views during testing. So more advanced processing is necessary.

One possibility is to exploit the relation between occluding and occluded object, which for natural scenes usually shows rather systematic patterns. The detection approach in [9] first searches for larger and thus more easily detectable objects and later exploits spatial relations to improve the detection of smaller, more difficult ones. This concept can be transferred to the occlusion problem. So one could train special detectors for different types of occlusion and exploit their mutual response characteristic in a scene.

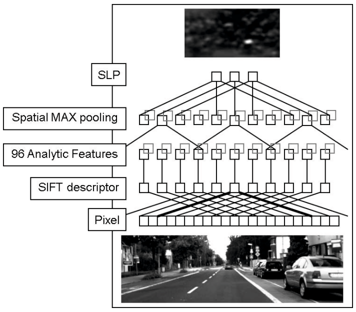
Other approaches that make use of object-object relations are presented in [10,4], where Markov-Random-Fields are used to infer if neighboring features are consistent with a single detected instance or have to be assigned to different ones. In this way both approaches can reason about relative depth of objects and produce a coarse segmentation. However, in this paper we propose a more directed search for occluded objects, instead of using such demanding iterative processing over the full scene.

Also convolutional neural architectures were recently applied with great success on current recognition [1] and segmentation benchmarks [8]. The problem of occlusion, however, was not actively treated in these models so far. We propose a particular training procedure for occluded and unoccluded detectors that can be applied for these architectures similarly.

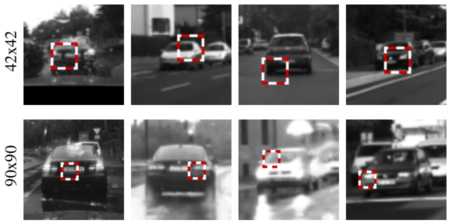
In Sec. 2 we outline how we adapted the analytic feature framework for detection tasks. After a short description of the traffic scene data used in our experiments, we evaluate the performance in Sec. 3. In Sec. 4 we propose a possible way to improve the detection of occluded cars and provide a first proof of concept on segmented car images, before drawing the conclusion in Sec. 5.

2 Adaptation of Analytic Feature Framework

We base our appearance-based detector on the real-time object identification framework in [5], which uses an attention mechanism to generate size normalized segments of the input object. Over the gray-scale segment first SIFT descriptors [7] are computed on a regular grid. Each descriptor is then matched to a set of 421 analytic features which are the result of the supervised selection process proposed in [5]. After this for each feature the global maximum is computed over the segment, in this way removing all spatial information. Finally a Single Layer



(a)



(b)

Fig. 1. (a) Analytic feature hierarchy. SIFT descriptors are computed on a regular grid and matched to 96 analytic features. After a local maximum filter per feature the SLP templates are used in a convolutional step. Maxima in the final response map denote possible car locations. (b) Some analytic features for two template sizes.

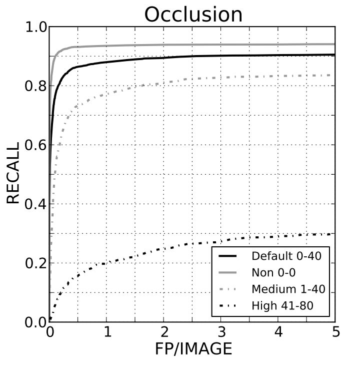
Perceptron (SLP) is used to separate the 126 objects in the 421-dimensional space. The approach is working robustly for full 3D rotation, even for untextured objects which are notoriously difficult for the standard SIFT approach [7].

The application of the existing framework for full scene object detection requires several adaptations. The rotation normalization of the SIFT descriptors is switched off because cars usually occur only upright. To speed up processing only 96 analytic features are used, where only features of the car class and not the background class were selected (see some examples in Fig. 1b). On the highest layer the local SLP template is shifted over the input image. This convolutional step is used to generate the car response map producing broad activation blobs for a car. The reason for this was the global maximum operation over features inside the template, which we had to replace with a local one to keep robustness against small translations. The resulting feature architecture is shown in Fig. 1a.

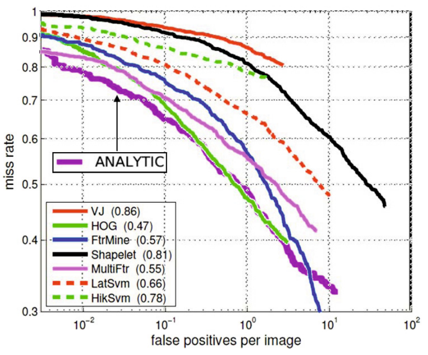
To deal with cars at different distances, we train analytic features and SLPs on three different segment sizes and use them on the largest image resolution, while the largest template is also used on successively reduced resolutions. This combined strategy improves detection performance because no compromise between minimal template size and most discriminative template size needs to be found, which is a common drawback of other detection approaches.

3 Detection Results

We decided to use the detection framework proposed in this paper to locate cars in real world traffic scenes. For this we equipped a car with a stereo camera and acquired different streams with a total length of 45 minutes covering different weather conditions (sunny, rainy, overcast) and scene types (city, rural, industry, highway). For one frame per second we labeled typical traffic participants with a ROI and also roughly estimated their percentage of occlusion.



(a)



(b)

Fig. 2. (a) ROC for our car detection scenario. The performance decreases strongly with the percentage of the cars' occlusion. (b) ROC for pedestrian benchmark (unoccluded, 50 pixels or taller). The analytic approach is on par with state-of-the-art approaches.

To evaluate the car detection performance we split the image streams into chunks of 30 seconds and used the odd chunks for the training of analytic features and SLP templates and the even chunks for testing. We decided to exclude car ROIs, whose width was more than twice their height (roughly 10% of the data), from training. In this way the use of smaller and square SLP templates was sufficient. So the templates were trained on segments of 42x42, 66x66, and 90x90 pixels (plus 18 pixels border at each side), into which the car was centered. Initially, the SLPs were trained to separate few thousand segments of unoccluded cars from a larger set of randomly chosen non-car segments. During 5 bootstrapping steps more negative examples were generated.

Please note that we used the disparity information available for our image data to reject implausible car candidates with simple hand-tuned rules on height-above-ground and physical-size. Finally, a local competition removed further weak hypotheses if they had a too strong overlap with more confident ones. For the input images of size 800x600 the GPU implementation of our framework runs with 10 frames per second on a mobile Geforce GTX580M.

The results for our car scenes are shown in Fig. 2a. We excluded cars with a height below 35 pixels and used the common 50% mutual overlap criterion between labels and detections. The curves reveal a strong dependency on the percentage of the cars' occlusion. For a false positive per image rate of 0.1 we get 70% of the cars with an occlusion between 0-40%. This pure detection performance is usually sufficient for a system that applies some kind of temporal integration (tracking). However for higher percentages of occlusion the recall drops severely, which can no longer be compensated at system level. In the next section we propose a special approach for detection of occluded cars.

Table 1. Counts of car occluders and occluded car parts for the ground truth data. In total 8796 out of 15514 cars are occluded, most of them by other cars.

Occluding object	#	Occluding object	#	Occluded part	#
Another car	7061	Pedestrian	70	Left	3730
Image border	2137	Traffic sign	31	Right	3124
Motor bike	82	Other/non-labeled	1125	Middle (only)	90

To get a comparison with state-of-the-art methods we decided to apply our framework also to the pedestrian detection benchmark proposed in [3]. We evaluated the performance by doing 6-fold crossvalidation over the 6 streams and averaged the results. In contrast to this the competitors in Fig. 2b used all streams for testing and trained on other pedestrian data each. So the results are not 100% comparable. However, because the streams are quite different from each other and the overall label quality is not that high, there is no obvious advantage in using the streams for training. So Fig. 2b roughly shows that we are at least competitive to the popular HOG approach [2]. The main conceptual difference to HOG is that it applies an SVM directly on top of the local gradient histograms, while we use an additional projection to the analytic features and a simple SLP as classifier. Please note that because of the missing stereo information only the mutual overlap heuristic was used here.

4 Occlusion Handling Using Object-Object Relations

In this section we propose a method to increase the detection performance for occluded cars. Following our discussion in the introduction, for this we like to exploit object-object relations. Taking into account that most cars are occluded by other cars, as revealed by the analysis of the ground truth shown in Tab. 1, we decided to implement following simple strategy: In order not to decrease performance for unoccluded cars we will train a special classifier on occluded cars. This new classifier will be applied in the vicinity of already detected cars only. In a scene these initial car hypotheses are generated by the detection framework described before using the classifiers trained on unoccluded cars. The conditional application rule is necessary to avoid a strong increase of false positives (FP), which would be the result of the independent usage of both classifiers.

For a fast proof of concept, we decided to first test this strategy on segmented car and non-car views. So we generated data pairs i, each having a **F**oreground segment $\boldsymbol{F}_i$ containing the occluder and the corresponding **B**ackground segment $\boldsymbol{B}_i$ containing something occluded. We refer to the set of all foreground/background segments with $\boldsymbol{F} = \{\boldsymbol{F}_i\}$ and $\boldsymbol{B} = \{\boldsymbol{B}_i\}$ respectively. The types of pairs that mimic all possible constellations in a scene are described in Fig. 3.

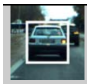
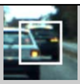
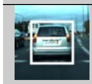
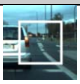
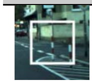
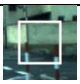
$\boldsymbol{F}_i$	$\boldsymbol{B}_i$	
 car	 car	**1. Car occluding car:** The occluding car is inserted as positive example to $\boldsymbol{F}$ and the occluded car as positive to $\boldsymbol{B}$.
 car	 no car	**2. Car not occluding car:** The car is put as positive to $\boldsymbol{F}$ and a randomly chosen, car-free region in its vicinity as negative to $\boldsymbol{B}$.
 no car	 no car	**3. Car-free pairs:** In a real scene the initial detector will produce false positives. The FPs of our detection framework are inserted as negatives to $\boldsymbol{F}$ and a randomly chosen, car-free region next to each FP as negative example to $\boldsymbol{B}$.

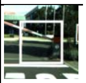
Fig. 3. Segment pair types. Each pair has a foreground segment $\boldsymbol{F}_i$ and a background segment $\boldsymbol{B}_i$. The positive examples (in gray) are generated from ground truth. For simplification we only use samples with occlusion at the left side and mirror examples with right occlusion to get more data. The classifier looks at the marked inner 42x42 pixel region of the segments into which the cars are fitted.

For the segment scenario we simply use the already trained SLP for a car size of 42x42 as initial classifier, which will be referred to as C_{Std}, and train the new classifier C_{Occ} on the background segments $\boldsymbol{B}$ of same size using cars with an occlusion up to 80%. The logic C_{Com} combines C_{Std} and C_{Occ} in a conditional manner and predicts the labels $L_{\boldsymbol{F}_i}$ and $L_{\boldsymbol{B}_i}$ for each pair using the code:

```
$L_{\boldsymbol{F}_i}$ = 'no car'
$L_{\boldsymbol{B}_i}$ = 'no car'
if $C_{\mathrm{Std}}(\boldsymbol{F}_i) \geq T_{\mathrm{Std}}$ then
    $L_{\boldsymbol{F}_i}$ = 'car'
    if $C_{\mathrm{Std}}(\boldsymbol{B}_i) \geq T_{\mathrm{Std}}$ then
        $L_{\boldsymbol{B}_i}$ = 'car'
    else if $C_{\mathrm{Occ}}(\boldsymbol{B}_i) \geq T_{\mathrm{Occ}}$ then
        $L_{\boldsymbol{B}_i}$ = 'car'
```

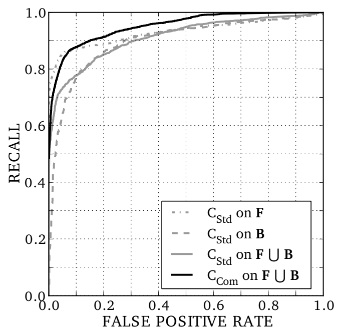
So $\boldsymbol{B}_i$ is predicted as car, if either C_{Std} or the new classifier C_{Occ} reaches its corresponding threshold, **and** only if a car was found in the foreground segment $\boldsymbol{F}_i$ already. We have chosen a low FPR of 0.1 to avoid disturbing false alarms, which goes along with a recall of 0.84 ($C_{\mathrm{Std}} = 0.37$, $C_{\mathrm{Occ}} = 0.34$). Some classification examples are shown in Fig. 4.

To show the benefit of the combined logic, in Fig. 5a we compare the performance of C_{Com} with C_{Std} and in Fig. 5b with C_{Occ}. Please note that the combined approach depends on the two thresholds T_{Std} and T_{Occ}, and thus we have to evaluate the performance of C_{Com} for each combination of them. However, most of the resulting points in the ROC curve are dominated by a small set of other points. We reduced the amount of the points and only show in the plot a so called Pareto Front, which describes the set of optimal combinations.

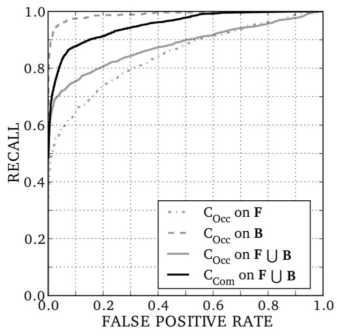
$\boldsymbol{F}_i$						
Ground truth	car	car	car	no car	car	no car
$c_{\text{Std}} \geq \tau_{\text{Std}}$	yes	yes	no	no	yes	yes
c_{Com} result	TP	TP	FN	TN	TP	FP
$\boldsymbol{B}_i$						
Ground truth	car	car	car	no car	no car	no car
$c_{\text{Std}} \geq \tau_{\text{Std}}$	yes	no	no	no	no	yes
$c_{\text{Occ}} \geq \tau_{\text{Occ}}$	yes	yes	yes	yes	yes	no
c_{Com} result	TP	TP	FN	TN	FP	FP

TP - true positive
TN - true negative
FP - false positive
FN - false negative

Fig. 4. Pair classification examples. For each foreground sample $\boldsymbol{F}_i$ we show the ground truth label, the decision of C_{Std}, and the result of the combined approach C_{Com}. For $\boldsymbol{B}_i$ we additionally show the decision of C_{Occ} because C_{Com} depends on both classifiers and on the result for $\boldsymbol{F}_i$. Dark gray is used for 'no car' labels and responses below threshold, light gray for the opposite. The conditional logic can correct FPs of C_{Occ} (4^{th} column), but in rare cases also prevents correct detections (3^{rd} column). The classifiers look at the marked inner 42x42 region.



(a)



(b)

Fig. 5. Comparison of C_{Com} with C_{Std} and C_{Occ}. (a) C_{Std} shows a good performance for the foreground segments $\boldsymbol{F}$ while the result for the occluded cars $\boldsymbol{B}$ is significantly weaker. On the combined data set $\boldsymbol{F} \cup \boldsymbol{B}$, C_{Com} is in general much better than C_{Std}. (b) C_{Occ} performs very good on $\boldsymbol{B}$ but has strong problems on the unfamiliar occluders $\boldsymbol{F}$. C_{Com} also clearly outperforms C_{Occ} on $\boldsymbol{F} \cup \boldsymbol{B}$.

Figure 5a confirms again that C_{Std} can cope substantially better with the familiar foreground segments $\boldsymbol{F}$ than with the occluded segments in $\boldsymbol{B}$. On the combined data set $\boldsymbol{F} \cup \boldsymbol{B}$ the classification has some intermediate quality but is clearly dominated by C_{Com}. For example, at a recall of 0.8 C_{Std} has a false positive rate of 0.13, while that of the combined curve is 0.04. This is a threefold reduction in the number of false positives.

In Figure 5b, C_{Occ} shows a very good performance on the occluded segments $\boldsymbol{B}$, for which it was trained. However, the performance for the unoccluded cars in $\boldsymbol{F}$ is much worse. One reason for this might be that C_{Occ} specialized too strongly on the edge that is caused by the occluder and which is not present in the unoccluded examples. Also in comparison to C_{Occ}, C_{Com} shows a substantially improved performance on the full data ensemble.

5 Conclusion

In this paper we presented a new object detection framework, which is based on the analytic feature representation originally proposed for object identification. We have shown the detection performance of the approach on a public pedestrian detection benchmark and evaluated on our own benchmark how strong occlusion effects the detection of cars. Motivated by these results and by an analysis of typical occlusion causes we proposed a new combination of detectors that takes the occlusion of cars by other cars into account. In a pre-study we successfully showed the benefit of this approach on segmented car views. The next step is to exploit the same principle also in full-scene detection and to other object relations, like persons occluded by other objects, or more general occlusion cases.

References

1. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. Neural Computation 22(12), 3207–3220 (2010)
2. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: CVPR, pp. 886–893 (2005)
3. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian Detection: A Benchmark. In: CVPR, pp. 304–311 (2009)
4. Gao, T., Packer, B., Koller, D.: A Segmentation-aware Object Detection Model with Occlusion Handling. In: CVPR, pp. 1361–1368 (2011)
5. Hasler, S., Wersing, H., Kirstein, S., Körner, E.: Large-scale real-time object identification based on analytic features. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009, Part II. LNCS, vol. 5769, pp. 663–672. Springer, Heidelberg (2009)
6. Leibe, B., Schiele, B.: Interleaved Object Categorization and Segmentation. In: BMVC, pp. 759–768 (2003)
7. Lowe, D.G.: Distinctive Image Features from Scale-invariant Keypoints. IJCV 60(2), 91–110 (2004)
8. Schulz, H., Behnke, S.: Learning Object-Class Segmentation with Convolutional Neural Networks. In: ESANN, pp. 151–156 (2012)
9. Torralba, A., Murphy, K.P., Freeman, W.T.: Contextual Models for Object Detection Using Boosted Random Fields. In: ICIP, pp. 653–656 (2011)
10. Winn, J., Shotton, J.D.J.: The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects. In: CVPR, pp. 37–44 (2006)
11. Yi-Hsin, L., Tz-Huan, H., Tsai, A., Wen-Kai, L., Jui-Yang, T., Yung-Yu, C.: Pedestrian Detection in Images by Integrating Heterogeneous Detectors. In: ICS, pp. 252–257 (2010)

Boltzmann Machines for Image Denoising

KyungHyun Cho

Department of Information and Computer Science
Aalto University School of Science, Finland
firstname.lastname@aalto.fi

Abstract. Image denoising based on a probabilistic model of local image patches has been employed by various researchers, and recently a deep denoising autoencoder has been proposed in [2] and [17] as a good model for this. In this paper, we propose that another popular family of models in the field of *deep learning*, called Boltzmann machines, can perform image denoising as well as, or in certain cases of high level of noise, better than denoising autoencoders. We empirically evaluate these two models on three different sets of images with different types and levels of noise. The experiments confirmed our claim and revealed that the denoising performance can be improved by adding more hidden layers, especially when the level of noise is high.

Keywords: Image Denoising, Deep Learning, Restricted Boltzmann Machine, Deep Boltzmann Machine.

1 Introduction

Numerous approaches based on machine learning have been proposed for image denoising tasks over time. A dominant approach has been to perform denoising based on local statistics of image patches. Under this approach, small image patches from a larger noisy image are denoised and combined afterward to form a clean image.

For instance, in [9] independent component analysis (ICA) was used to estimate a dictionary of sparse elements and compute the sparse code of image patches. Subsequently, a shrinkage nonlinear function is applied to the estimated sparse code elements to suppress those elements with small absolute magnitude. These shrunk sparse codes are then used to reconstruct a noise-free image patch. More recently, it was shown in [5] that sparse overcomplete representation may be more useful in denoising images.

In essence, these approaches build a probabilistic model of natural image patches using a single layer of sparse latent variables. The posterior distribution of each noisy patch is either exactly computed or estimated, and the noise-free patch is reconstructed as an expectation of a conditional distribution over the posterior distribution.

Some researchers have proposed very recently to utilize a model that has more than one layers of latent variables for image denoising. It was shown in [2] that a deep denoising autoencoder [16] that learns a mapping from a noisy image patch to its corresponding clean version, can perform as good as the state-of-the-art denoising methods. Similarly, a variant of a stacked sparse denoising autoencoder that is more effective in image denoising was recently proposed in [17].

© Springer-Verlag Berlin Heidelberg 2013

Along this line of research, we propose yet another type of deep neural networks for image denoising, in this paper. A Gaussian-Bernoulli restricted Boltzmann machines (GRBM) [6] and deep Boltzmann machines (GDBM) [13] are empirically shown to perform well in image denoising, compared to stacked denoising autoencoders. Furthermore, we evaluate the effect of the number of hidden layers of both Boltzmann machines and denoising autoencoders. The empirical evaluation is conducted using different noise types and levels on three different sets of images.

2 Deep Neural Networks

We start by briefly describing Boltzmann machines and denoising autoencoders which have become increasingly popular in the field of machine learning.

2.1 Boltzmann Machines

Originally proposed in 1980s, a Boltzmann machine (BM) [1] and especially its structural constrained version, a restricted Boltzmann machine (RBM) [14] have become increasingly important in machine learning since it was shown that a deep multi-layer perceptron can be trained easily by stacking RBMs on top of each other [6]. More recently, another variant of a BM, called a deep Boltzmann machine (DBM), has been proposed and shown to outperform other conventional machine learning methods in many tasks (see, e.g., [12]).

We first describe a Gaussian-Bernoulli DBM (GDBM) that has L layers of binary hidden units and a single layer of Gaussian visible units. A GDBM is defined by its energy function

$$\begin{aligned}-E(\mathbf{v},\mathbf{h}\mid\boldsymbol{\theta}) =& \sum_i -\frac{(v_i-b_i)^2}{2\sigma^2} + \sum_{i,j}\frac{v_i}{\sigma^2}h_j^{(1)}w_{i,j} + \sum_j h_j^{(1)}c_j^{(1)} \\ &+ \sum_{l=2}^{L}\left(\sum_j h_j^{(l)}c_j^{(l)} + \sum_{j,k} h_j^{(l)}h_k^{(l+1)}u_{j,k}^{(l)}\right), \end{aligned} \tag{1}$$

where $\mathbf{v}=[v_i]_{i=1\dots N_v}$ and $\mathbf{h}^{(l)}=\left[h_j^{(l)}\right]_{j=1\dots N_l}$ are N_v Gaussian visible units and N_l binary hidden units in the l-th hidden layer. $\mathbf{W}=[w_{i,j}]$ is the set of weights between the visible neurons and the first layer hidden neurons, while $\mathbf{U}^{(l)}=\left[u_{j,k}^{(l)}\right]$ is the set of weights between the l-th and $l+1$-th hidden neurons. σ^2 is the shared variance of the conditional distribution of v_i given the hidden units.

With the energy function, a GDBM can assign a probability to each state vector $\mathbf{x}=[\mathbf{v};\mathbf{h}^{(1)};\cdots;\mathbf{h}^{(L)}]$ using a Boltzmann distribution: $p(\mathbf{x}\mid\boldsymbol{\theta})=\exp\{-E(\mathbf{x}\mid\boldsymbol{\theta})\}/Z(\boldsymbol{\theta})$. Then, the parameters can be learned by maximizing the log-likelihood

$$\mathcal{L}=\sum_{n=1}^{N}\log\sum_{\mathbf{h}}p(\mathbf{v}^{(n)},\mathbf{h}\mid\boldsymbol{\theta})$$

given N training samples $\{\mathbf{v}^{(n)}\}_{n=1,\dots,N}$, where $\mathbf{h}=\left[\mathbf{h}^{(1)};\cdots;\mathbf{h}^{(L)}\right]$.

Although the update rules based on the gradients of the log-likelihood function are well defined, it is intractable to exactly compute them. Hence, an approach that uses variational approximation together with Markov chain Monte Carlo (MCMC) sampling was proposed in [13]. This approach is often used together with a pretraining algorithm [13,4]. In this paper, we initialized GDBMs using the two-stage pretraining algorithm recently proposed in [4].

A Gaussian-Bernoulli RBM (GRBM) is a special case of a GDBM, where the number of hidden layers is restricted to one, $L = 1$. Unlike GDBMs, it is possible to compute the posterior distribution over the hidden units exactly and tractably by

$$p(h_j = 1 \mid \mathbf{v}, \boldsymbol{\theta}) = f\left(\sum_i w_{ij}\frac{v_i}{\sigma^2} + c_j\right), \tag{2}$$

where f is a logistic sigmoid function, and we dropped the superscript (1) from h_j for simplicity.

This removes the need for the variational approximation in computing the gradient and allows an efficient block Gibbs sampling. Based on this property many fast approximate algorithms for training GRBMs, such as contrastive divergence [7], have been proposed.

2.2 Denoising Autoencoders

A denoising autoencoder (DAE) [16] is a special form of multi-layer perceptron network with $2L-1$ hidden layers and $L-1$ sets of (tied) weights. A DAE tries to learn a network that denoises an explicitly corrupted input vector by minimizing the following cost function:

$$\sum_{n=1}^{N}\left\|\mathbf{W}g^{(1)} \circ \cdots \circ g^{(L-1)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}\left(\eta(\mathbf{v}^{(n)})\right) - \mathbf{v}^{(n)}\right\|^2, \tag{3}$$

where $f^{(l)} = \phi(\mathbf{W}^{(l)\top}\mathbf{h}^{(l-1)})$ and $g^{(l)} = \phi(\mathbf{W}^{(l)}\mathbf{h}^{(2L-l)})$ are, respectively, encoding and decoding functions for l-th layer with a component-wise nonlinearity function ϕ. η stochastically adds noise to an input vector $\mathbf{x}$ at each update step. $\mathbf{W}^{(l)}$ is the weights between the l-th and $(l+1)$-th layers and is shared by the encoder and decoder.

3 Image Denoising

Let us define a set of N binary matrices $\mathbf{D}_n \in \mathbb{R}^{p\times d}$ that extract a set of small image patches given a large, whole image $\mathbf{x} \in \mathbb{R}^d$, where $d = wh$ is the product of the width w and the height h of the image and p is the size of image patches. Then, an image can be denoised by

$$\mathbf{x} = \left(\sum_{n=1}^{N}\mathbf{D}_n^\top r_{\boldsymbol{\theta}}(\mathbf{D}_n\tilde{\mathbf{x}})\right) \oslash \left(\sum_{n=1}^{N}\mathbf{D}_n^\top\mathbf{D}_n\mathbf{1}\right), \tag{4}$$

where $\oslash$ is an element-wise division and $\mathbf{1}$ is a vector of ones. $r_{\boldsymbol{\theta}}(\cdot)$ is an image denoising function, parameterized by $\boldsymbol{\theta}$, that denoises each image patch $\mathbf{D}_n\mathbf{x}$.

In short, Eq. (4) extracts and denoises image patches from the input image. Then, it combines them by taking an average of those overlapping pixels.

One of the popular choices for $r_{\boldsymbol{\theta}}(\cdot)$ has been to construct a probabilistic model with a set of latent variables that describe natural image patches (see, e.g., [8,5]). Under this approach denoising can be considered as a two-step reconstruction. First, the posterior distribution over the latent variables is computed given an image patch. Based on that, the conditional distribution, or its mean, over the visible units is computed and used as a denoised image patch.

3.1 Boltzmann Machines

We consider a BM with a set of Gaussian visible units $\mathbf{v}$ that correspond to the pixels of an image patch and a set of binary hidden units $\mathbf{h}$. Then, the goal of denoising can be written as

$$p(\mathbf{v} \mid \tilde{\mathbf{v}}) = \sum_{\mathbf{h}} p(\mathbf{v} \mid \mathbf{h}) p(\mathbf{h} \mid \tilde{\mathbf{v}}) = \mathbb{E}_{\mathbf{h}|\tilde{\mathbf{v}}} \left[p(\mathbf{v} \mid \mathbf{h}) \right], \tag{5}$$

where $\tilde{\mathbf{v}}$ is a noisy input patch. In other words, we find a mean of the conditional distribution of the visible units with respect to the posterior distribution over the hidden units given the visible units fixed to the corrupted input image patch.

However, since taking the expectation over the posterior distribution is usually not tractable, it is often easier and faster to approximate it. We approximate the marginal conditional distribution in (5) with $p(\mathbf{v} \mid \tilde{\mathbf{v}}) \approx p(\mathbf{v} \mid \mathbf{h})Q(\mathbf{h})$, where $Q(\mathbf{h})$, parameterized by $\boldsymbol{\mu} = \mathbb{E}_{Q(\mathbf{h})}[\mathbf{h}]$, is a fully-factorial (approximate) posterior distribution $p(\mathbf{h} \mid \tilde{\mathbf{v}})$.

Given a noisy image patch $\tilde{\mathbf{v}}$, following this approach, a GRBM reconstructs a noise-free patch by

$$\hat{v}_i = \sum_{j=1}^{N_h} w_{ij} \mathbb{E}\left[\mathbf{h} \mid \tilde{\mathbf{v}}\right] + b_i.$$

The conditional distribution over the hidden units can be computed exactly from Eq. (2).

Unlike a GRBM, the posterior distribution of the hidden units of a GDBM is neither tractably computable nor has an analytical form. Instead, we use a fully-factorial *approximate* posterior $Q(\mathbf{h}) = \prod_{l=1}^{L} \prod_j \mu_j^l$, where the parameters $\mu_j^{(l)}$'s can be estimated by maximizing the variational lower-bound [13].

Once the variational parameters are converged, a GDBM reconstructs a noise-free patch by

$$\hat{v}_i = \sum_{j=1}^{N_l} w_{ij} \mu_j^{(1)} + b_i.$$

The convergence of the variational parameters may take too much time in practice. Hence, in the experiments, we initialize the variational parameters by the feed-forward propagation using the doubled weights [13] and performing the fixed-point update for at most five iterations only.

Table 1. Descriptions of the test image sets

Set	# of all images	# of color images	Min. Size	Max. Size
Textures	64	0	512×512	1024×1024
Aerials	38	37	512×512	2250×2250
Miscellaneous	44	16	256×256	1024×1024

3.2 Denoising Autoencoders

An encoder part of a DAE can be considered as performing an approximate inference of a fully-factorial posterior distribution of top-layer hidden units, i.e. a bottleneck, given an input image patch [16]. Hence, a similar approach can be applied to DAEs.

Firstly, the variational parameters $\boldsymbol{\mu}^{(L)}$ of the fully-factorial posterior distribution $Q(\mathbf{h}^{(L)}) = \prod_j \mu_j^{(L)}$ are computed by

$$\boldsymbol{\mu}^{(L)} = f^{(L-1)} \circ \cdots \circ f^{(1)} (\tilde{\mathbf{v}}) .$$

Then, the denoised image patch can be reconstructed simply by propagating the variational parameters through the decoding nonlinearity functions $g^{(l)}$ such that

$$\hat{\mathbf{v}} = g^{(1)} \circ \cdots \circ g^{(L-1)} \left(\boldsymbol{\mu}^{(L)} \right) .$$

4 Experiments

In the experiment, we test both GRBMs and GDBMs together with DAEs having varying numbers of hidden layers. These models are compared to each other on a *blind* image denoising task, where no prior knowledge about target images nor the type or level of noise is known when the models are trained. In other words, no separate training was done for different types or levels of noise injected to the test images. Unlike this, for instance, in [17] each DAE was trained specifically for the target noise level and type by changing $\eta(\cdot)$ accordingly.

4.1 Datasets

We used three sets of images, *textures*, *aerials* and *miscellaneous*, from the USC-SIPI Image Database[1] as test images. Tab. 1 lists the details of the image sets.

These datasets are, in terms of contents and properties of images, very different from each other. For instance, most of the images in the texture set have highly repetitive patterns that are not present in the images in the other two sets. Most images in the aerials set have simultaneously both coarse and fine structures. Also, the sizes of the images vary quite a lot across the test sets and across the images in each set.

As we are aiming to evaluate the performance of denoising a very general image, we used a large separate data set of natural image patches to train the models. We used a

[1] `http://sipi.usc.edu/database/`

large number of image patches randomly of sizes 4×4, 8×8 and 16×16 extracted from CIFAR-10 dataset [10]. The same set of experiments was run with the models trained on the patches from the Berkeley Segmentation Dataset [11], and the similar results were observed.

4.2 Settings

We tried three different depth settings for both Boltzmann machines and denoising autoencoders; a single, two and four hidden layers. Each hidden layer had the same number of hidden units which was the constant factor 5 multiplied by the number of pixels in an image patch, as suggested in [17].

We denote Boltzmann machines with one, two and four hidden layers by GRBM, GDBM(2) and GDBM(4), respectively. Denoising autoencoders are denoted by DAE, DAE(2) and DAE(4), respectively.

The GRBMs were trained using the enhanced gradient [3] and persistent contrastive divergence (PCD) [15]. The GDBMs were trained by PCD after initializing the parameters with a two-stage pretraining algorithm [4]. DAEs were trained by a stochastic backpropagation, and when there were more than one hidden layers, we pretrained each pair of consecutive layers as a single-layer DAE with sparsity target set to 0.1.

Two types of noise have been tested; white Gaussian and salt-and-pepper. White Gaussian noise simply adds zero-mean normal random value with a predefined variance to each image pixel, while salt-and-pepper noise sets a randomly chosen subset of pixels to either black or white. Three different noise levels (0.1, 0.2 and 0.4) were tested. In the case of white Gaussian noise, they were used as standard deviations, and in the case of salt-and-pepper noise, as a noise probability.

After noise was injected, each image was preprocessed by pixel-wise adaptive Wiener filtering, following the approach of [9]. The width and height of the pixel neighborhood were chosen to be small enough (3×3) to avoid removing too much detail.

4.3 Results and Analysis

In Fig. 1, the performances of all the tested models trained on 8×8 image patches, measured by the peak signal-to-noise ratio (PSNR), are presented. Those trained on the patches of different sizes showed similar trend, and they are omitted here.

Interestingly, the GRBMs consistently performed comparably to much deeper DAEs in most cases regardless of the level of noise. This indirectly suggests that learning a good generative model might be important in image denoising. Considering that the number of parameters of the GRBM is, for instance, only half compared to DAE(2), training a model in a fully generative manner helps learning more compact representation of natural image patches that are more suitable for image denoising.

However, the GDBMs which are essentially deeper version of the GRBM were only able to outperform the other models in the high noise regime. In the cases of the aerials and miscellaneous sets, in the low noise regime, the GDBMs lag behind all the other models. A possible explanation for this rather poor performance of the GDBMs in the low noise regime is that the posterior distribution had to be approximated, whereas it

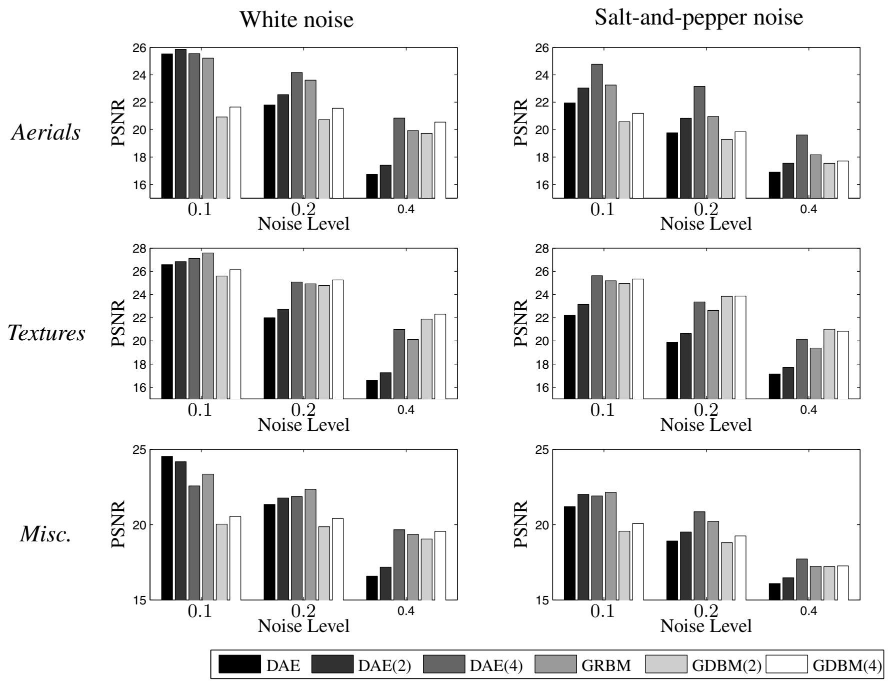


Fig. 1. The median PSNRs of grayscale images corrupted by different types and levels of noise

was computed exactly in the case of GRBMs. A better approximation strategy might resolve this problem.

An important observation is that the deeper models significantly outperformed their corresponding shallower models as the level of injected noise grew. In other words, the power of the deep models became more evident as the difficulty of the task increased.

5 Discussion

In this paper, we proposed that, in addition to DAEs, Boltzmann machines (BM) can also be used efficiently for denoising images. Boltzmann machines and DAEs were empirically evaluated against each other in the *blind* image denoising task where no prior knowledge about target images and their level of corruption was assumed.

The experimental results showed that Boltzmann machines (BM) are good, potential alternatives to DAEs. BMs and DAEs performed comparably to each other in the low noise regime, while BMs were able to, in many cases, outperform DAEs when the level of injected noise was high. This suggests that BMs may be more robust to noise than DAEs are.

More careful look at the experimental results clearly showed that, in the case of DAEs, hidden layers do improve performance, especially when the level of noise is high. This did not always apply to BMs, where we found that the GRBMs outperformed, or performed as well as, the GDBMs in many cases. Regardlessly, in the high noise regime, it was always beneficial to have more hidden layers, even for BMs.

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science 9, 147–169 (1985)
2. Burger, H., Schuler, C., Harmeling, S.: Image denoising: Can plain neural networks compete with bm3d? In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2392–2399 (June 2012)
3. Cho, K., Raiko, T., Ilin, A.: Enhanced gradient for training restricted Boltzmann machines. Neural Computation 25(3), 805–831 (2013)
4. Cho, K., Raiko, T., Ilin, A., Karhunen, J.: A Two-Stage Pretraining Algorithm for Deep Boltzmann Machines. In: NIPS 2012 Workshop on Deep Learning and Unsupervised Feature Learning, Lake Tahoe (December 2012)
5. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing 15(12), 3736–3745 (2006)
6. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
7. Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771–1800 (2002)
8. Hyvärinen, A.: Fast and robust fixed-point algorithms for independent component analysis. IEEE Transactions on Neural Networks 10(3), 626–634 (1999)
9. Hyvärinen, A., Hoyer, P., Oja, E.: Image denoising by sparse code shrinkage. In: Intelligent Signal Processing. IEEE Press (1999)
10. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., Computer Science Department, University of Toronto (2009)
11. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision, vol. 2, pp. 416–423 (July 2001)
12. Salakhutdinov, R., Hinton, G.: An efficient learning procedure for deep Boltzmann machines. Neural Computation 24, 1967–2006 (2012)
13. Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: Proc. of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2009), pp. 448–455 (2009)
14. Smolensky, P.: Information processing in dynamical systems: foundations of harmony theory. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations, vol. 1, pp. 194–281. MIT Press, Cambridge (1986)
15. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proc. of the 25th Int. Conf. on Machine Learning (ICML 2008), pp. 1064–1071. ACM, New York (2008)
16. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11, 3371–3408 (2010)
17. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: Bartlett, P., Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 25, pp. 350–358 (2012)

Comparison on Late Fusion Methods of Low Level Features for Content Based Image Retrieval

Nikolay N. Neshov

Department of Radio Communications and Video Technologies
Technical University of Sofia, Bul. Kl. Ohridsky 8, Sofia 1797, Bulgaria
nneshov@tu-sofia.bg

Abstract. Finding the right feature for image representation is an important key to attaining successful Content Based Image Retrieval (CBIR) system. This choice depends on the content of images to be searched. Today's real world image databases are heterogeneous and consist of images that can be described appropriately using different feature types. One approach to deal with this is to utilize late fusion methods. That is, the CBIR system must be able to fuse multiple results produced by each feature. In this paper by experimental comparison of output results achieved from eleven low-level features over three image databases an appropriate several sets of features are selected and five late fusion methods are applied over each set. By analysis of the results for all methods it has been shown which ones reach the best performances and stable retrieval accuracy among the investigated image databases and sets of features.

Keywords: Content Based Image Retrieval, Late Fusion, LIRe.

1 Introduction

With the rapid development of the Internet and amount of image capturing devices such as scanners and digital cameras, the size of digital image databases is increasing dramatically every day. Users and experts from various domains (crime prevention, medicine, architecture, publishing, etc) need effective and efficient CBIR systems for image browsing and searching. CBIR is well-known field of research that is engaged in the organization of digital image archives by their visual content. That is, the images are indexed using their low-level features (color, texture, shape) and essentially no high-level semantic information (keywords, tags, labels, etc) takes place in the retrieval process. Having user manually enter additional description to the images is not always efficient because one may not capture every keyword that describes the image. Thus a system that can filter images based only on their visual content would provide better indexing and return more accurate results [1]. On the other hand visual information captured in the low-level features differs from the high-level semantic description that is near to users' intention when they are looking for a particular image. This is a common problem in the current CBIR systems and has been discussed in the literature under the name of "semantic gap". Thus disregarding high-level concept would lead to inaccurate retrieval results.

© Springer-Verlag Berlin Heidelberg 2013

It has been proposed and investigated a large number of methods for improvement of CBIR, but still no general satisfying approach has been found. Data fusion has the potential to increase retrieval performance relying on the assumption that the heterogeneity of multiple information sources allows cross-correction of some of the errors leading to better results [2]. The response of effective fusion method of several information sources should produce greater performance than that of its constituent parts.

There are two main approaches to fusion, namely: early fusion, where multiple image features are aggregated in a new feature prior indexing, and late fusion, where result lists produced from distinctive features are combined in a new ranked list.

In general, the late fusion approaches are most widely used and developed [2]. They can be accomplished in one of two manners: score-based, that takes into account the function of score (relevance, similarity or distance from the query) in each rank list and rank-based, that deals with the function of position in which the results are ordered in each rank list [3].

2 Related Work

There has been much work done in the area of combination of different sources of evidence for CBIR. For example, authors in [4] compared six late fusion methods for three features (Joint Composite Descriptor - JCD, Spatial Color Distribution Descriptor - SpCD and Brightness and Texture Directionality Histogram - BTDH). The JCD feature used in the analyses is based on early fusion approach combining Color and Edge Directivity Descriptor (CEDD) and Fuzzy Color and Texture Histogram (FCTH). It has been shown that JCD has better performance than both CEDD and FCTH. Each late fusion method tested outperforms single feature baselines. The authors in [5] combined six features to implement retrieval system. However their experiments are based only on simple combination of scores reached by each of the individual features. Dubey et al. [6] proposed a multi feature model for CBIR by combining Color Histogram, Color Moment, Texture and Edge features. The authors in [7] implemented two features. The first one is called Local Color Pixel Classification (LCPC) that incorporates both color and spatial information. In addition they proposed an extension of LCPC by including shape features. The authors in [8] developed a combination method which captures local color and texture features in a coarse segmentation framework of grids, and has a shape feature in terms of invariant moments computed on the edge image.

The goals of this paper are as follows: comparing the retrieval performance of eleven well known image features for CBIR and selecting the best ones; applying five late fusion methods using different sets of the best features; determining which fusion method and which set of features have the highest improvement of the retrieval performance with respect to the performance obtained by the best performing image feature over three popular image databases.

In contrast to some of the previous works [5], [6] and [9] the results presented in this article are measured over several different conditions i.e. sets of image features, late fusion methods and databases. This is of vital importance when attempting to build accurate CBIR system for heterogeneous image databases.

In order to compare the retrieval effectiveness of all tested runs we utilized the commonly used and well known parameter – Mean Average Precision (MAP) [9].

All represented algorithms are developed with the help of Lucene Image Retrieval (LIRe) [10], [11] - an open source Java library.

The paper is structured as follows: In Section 3 are presented the databases used in the experiments; in Section 4 is explained which low-level features are utilized for the combination algorithms; Section 5 gives more details about the late fusion methods that we investigated; Section 6 describes how the sets of features are composed along with the experimental results and Section 7 gives the conclusions of the current work.

3 Image Databases

For our experiments we used the following three image databases:

WANG [12] – Contains 1000 images manually divided in 10 classes of 100 images each. Each image from the database is used as query in the process of performance evaluation;

Uncompressed Color Image Dataset (UCID) [13] – Contains 1338 photos and a ground truth where 262 images (queries) from the database have been manually assigned to their similar ones;

Zurich Buildings Database for Image Based Recognition (ZuBuD) [14] – Consists of two parts: query and training part. The training part depicts 201 buildings in Zurich city. Each building is represented by five photos taken from different angles of view or weather conditions. Thus the training part contains 1005 images. The query part is formed from 115 images each of which represents one building from the training part.

4 Low Level Features

Based on our previous work of eleven low-level global features [9] (for UCID and ZuBuD databases) and the investigations done by M. Lux [11] (for WANG database)

Table 1. MAP of the features with the best retrieval performance for the three databases

	MAP, %		
Feature	WANG	UCID	ZuBuD
Joint Composite Descriptor (JCD)	**(50,95)**	**47,07**	**71,78**
Brightness Histogram (BH)	**48,44**	**39,23**	**(74,60)**
Auto Correlogram (AC)	**47,51**	**(53,5)**	**69,44**
Color Layout (CL)	43,88	**26,13**	**59,53**
DCT Histogram (DCTH)	**44,55**	25,9	1,49

we selected the top four features (per database) that reached the best retrieval performances. Table 1 summarizes the value of MAP for each feature. The highest four values for each database are shown in bold and the best value is surrounded by round brackets.

In Table 1 it can be seen that each database has its own best features. For example Color Layout is in the top four features for the UCID database but it is not for the WANG. Thus we have a total of five low-level features considered as a background in this work.

5 Late Fusion Methods

The scenario of all fusion methods tested in this work begins with the following baseline preprocessing. Using each examined feature j (Section 4) the CBIR system have to produce a list of images L_j arranged in ascending order with respect to their distance to the query image. Having all lists of images the goal of each fusion method is to create one final ranked list L_f .

Let D be the set of all N images in the database and let i be an image from this set ($i \in D$). Let's denote the score (distance to the query in our case) for image i in the arranged list L_j with $S_j(i)$ and its rank (position) with $R_j(i)$.

Now we will discuss how the particular fusion methods differs each other. As mentioned in Section 1 they can be classified as score-based and rank-based.

5.1 Score-Based Fusion Methods

First a final score $S_f(i)$ for each image needs to be calculated. Then all final scores are arranged in ascending order with respect to $S_f(i)$ and the final list L_f is generated. Three score-based fusion methods are implemented. The first one is:

CombSUM without normalization (CombSUM) – The scores of each list j are summed to obtain the final score [2]:

$$S_f(i) = \sum_{j=1}^{N_j} S_j(i) \ , \tag{1}$$

where N_j is the number of features (number of lists L_j) considered for combination;

In the method above the range of score values for each list L_j may vary. In this way the influence of result produced by each feature is not equal in the combination process. This imposes the necessity of normalization before computation of the final score. For this reason we applied the following two methods each of which includes linear normalization technique:

CombSUM with Min-Max normalization (CombSUM Min-Max) – Let $\overline{S}_j(i)$ be a normalized score for image i from the list j . Using Min-Max normalization the value of $\overline{S}_j(i)$ is computed as follows:

$$\overline{S}_j(i) = \frac{S_j(i) - S_j^{\min}}{S_j^{\max} - S_j^{\min}} \ , \tag{2}$$

where $S_j^{\max}$ and $S_j^{\min}$ are the lowest and highest scores found in the list L_j ;

CombSUM with Z-Score normalization (CombSUM Z-Score) – Z-Score normalizes each score to its number of standard deviations that it is away from the mean score. It is calculated like this:

$$\overline{S}_j(i) = \frac{S_j(i) - \mu}{\sigma} \ , \tag{3}$$

where μ is the average value of the un-normalized scores, and σ is the typical deviation [4].

For the two normalization methods described above, Equation 1 takes the form:

$$S_f(i) = \sum_{j=1}^{N_j} \overline{S}_j(i) \ . \tag{4}$$

5.2 Rank-Based Fusion Methods

Borda Count – The Borda Count method is based on democratic election strategies. The first (the most relevant) image in each ranked list L_j gets the maximum Borda Count points (votes). Each subsequent image gets one point less. Thus, the Borda Count points $BC_j(i)$ of image i in the list L_j are calculated like this:

$$BC_j(i) = N - R_j(i) \ , \tag{5}$$

where $R_j(i)$ takes integer value in the interval from 0 to $N-1$.

Next for each image the total BC points are calculated just like in CombSUM on ranks [2]:

$$BC(i) = \sum_{j=1}^{N_j} BC_j(i) \ , \tag{6}$$

Finally the images are classified in descending order according to the BC points;

Inverse Ranking Position (IRP) - The IRP merges ranked lists by calculation of IRP distance for each image $IRP(i)$:

$$IRP(i) = \frac{1}{\sum_{j=1}^{N_j} \frac{1}{R_j(i)}} \ . \tag{7}$$

where $R_j(i) \in [1 \div N]$. At the end of the process the images are ordered on the basis of their IRP distance [4].

6 Experimental Results

With aim to investigate appropriate sets of features for fusion we recall the results listed in Table 1 (Section 4). First we constructed all four possible sets by taking three of the best four features and one more set formed by all four features. Then each fusion method (Section 5) is applied over each of the five sets of features. The retrieval performance based on MAP for the particular databases is given in Table 2.

The best values with respect to the different fusion methods for each set of features are shown in bold. The name of the fusion method and the names of its composed features with the highest retrieval performance are also bolded. The value of MAP in this case is shown in round brackets. In square brackets are surrounded these values of MAP that are smaller than the MAP of the best individual feature.

Table 2. MAP [%] per fusion method and set of features, Average of MAP [%] and Standard Deviation of MAP per fusion method (last two columns) for each database (the best values are shown in bold)

WANG							
Set of Features / Fusion Method	JCD BH AC	JCD BH DCTH	**JCD AC DCTH**	BH AC DCTH	All Four Features	Aver. of MAP	St. Dev. of MAP
CombSUM	56,33	57,69	61,17	56,86	**61,23**	58,66	2,12
CombSUM Min-Max	56,44	57,94	61,18	60,19	60,71	59,29	1,81
CombSUM Z-Score	**56,86**	**58,04**	**(61,43)**	**60,59**	61,09	**59,6**	1,82
Borda Count	55,95	56	59,31	58,76	59,71	57,95	1,64
IRP	54,15	54,12	55,78	54,97	56,26	55,06	**0,86**
UCID							
Set of Features / Fusion Method	**AC JCD BH**	AC JCD CL	AC BH CL	JCD BH CL	All Four Features	Aver. of MAP	St. Dev. of MAP
CombSUM	**(60,59)**	**59,49**	**54,21**	[47,83]	**59,68**	**56,36**	4,82
CombSUM Min-Max	56,53	58,43	[52,83]	**[50,39]**	57,14	55,06	**2,99**
CombSUM Z-Score	58,2	59,39	54,11	[49,74]	57,63	55,81	3,51
Borda Count	54,56	[50,91]	[48,22]	[44,39]	[51,48]	49,91	3,42
IRP	56,46	56,75	[52,77]	[48,61]	56,39	54,2	3,15
ZuBuD							
Set of Features / Fusion Method	**BH JCD AC**	BH JCD CL	BH AC CL	JCD AC CL	All Four Features	Aver. of MAP	St. Dev. of MAP
CombSUM	83,95	[74,3]	79,64	**83,72**	83,95	81,1	3,78
CombSUM Min-Max	84,98	**79,45**	83,59	82,65	84,58	83,05	1,97
CombSUM Z-Score	**(85,26)**	78,8	**83,65**	82,87	**84,95**	**83,11**	2,32
Borda Count	81,61	[74,28]	77,68	77,9	80,04	78,3	2,48
IRP	81,48	77,76	79,89	79,22	81,75	80,02	**1,48**

In order to accomplish the comparison of the retrieval performance of the fusion methods regardless of the set of features the Average of MAP for each method is calculated. To estimate the stability of the retrieval performance the Standard Deviation of MAP is also computed (Table 2).

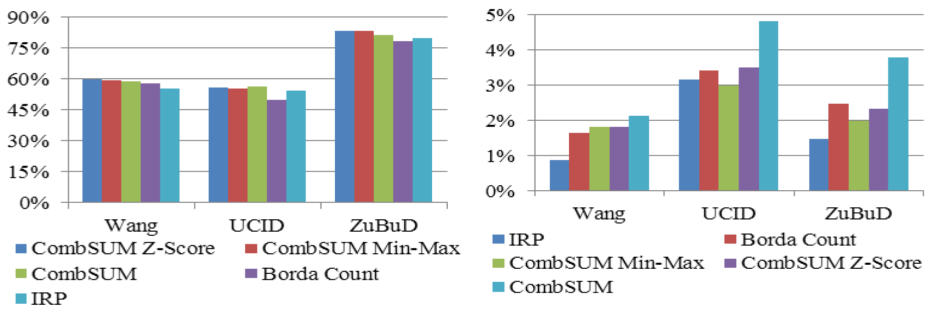


Fig. 1. Graphical comparison on Average of MAP (in left) and Standard Deviation of MAP (in right) per fusion method per database

Fig. 1 represents graphically the values of Average of MAP (left side) and Standard Deviation of MAP (right side) for each method and database. In Table 3 are displayed the highest values of MAP reached by the best fusion method and set of features compared to the best individual feature for each database. The last column shows the percentages of improvement of MAP.

Table 3. Comparison between the best baseline feature and the best fusion method

Database	The Best Feature – MAP, %	Fusion Method – MAP, %	Improvement, %
WANG	JCD – 50,95	CombSUM Z-Score – 61,43	20,57
UCID	AC – 53,5	CombSUM – 60,59	13,25
ZuBuD	BH – 74,6	CombSUM Z-Score – 85,26	14.29

In addition to our experimental scheme we also applied each late fusion method over the ranking lists produced using only two of the considered image features by taking each pair of all possible combinations (6 in total). In table 4 is shown the best case along with its corresponding fusion method for each database.

Table 4. The highest performance obtained by late fusion of two of the considered image features

Database	The Best Set of Two Features – Fusion Method	MAP, %
WANG	(JCD and DCTH) – CombSUM	57,57
UCID	(JCD and AC) – CombSUM Z-Score	60,46
ZuBuD	(JCD and AC) – CombSUM	83,77

From table 3 (third column) and table 4 (last column) can be seen that the best case of late fusion of three image features outperforms the best case of late fusion of two image features.

7 Conclusions and Future Work

From the experimental results it can be seen that in the most cases the fusion methods outperformed the best baseline features (Table 2). The highest achieved improvements of MAP along with the names of the fusion methods are shown in Table 3. CombSUM Z-Score is the best fusion method for WANG and ZuBuB databases and is on second place after CombSUM for UCID database. According to this study the best features for fusion are JCD and AC in combination with BH (for UCID and ZuBuD) and with JCD (for WANG).

IRP has the smallest value of Standard deviation of MAP for WANG and Zubud, but its retrieval performance is rather bad (Fig. 1). CombSUM has the worst deviation of MAP for all three databases (Table 2). Borda Count has the worst retrieval performance for UCID and ZuBuD. The maximum value of MAP using the set of four features for all databases is on second place after the method with the best performance.

Our studies demonstrate improvement of the retrieval effectiveness compared to some previous works. For example a combination technique presented in [15] (which is based on correlation analysis of features described for the WANG database) reaches a value of MAP equals to 55,7. In our experiments the best value of MAP for WANG database is 61,43 (Table 3).

There have been conducted various studies in the area of data fusion which attempt to understand when the combination of multiple systems (ranking lists in our case) works. Most of them conclude that the best time to combine a pair of systems is when one has reasonable performance and both return the same relevant documents and different non relevant documents [16]. Our experimental studies over three image databases show that the firs half of this hypothesis is likely to be correct for combinations of three systems in terms of CBIR. However from Table 2 it can be seen that there are only 12 cases (values of MAP in rounded brackets) of 75 cases (in total) where the performances are worse than those produced by the individual features. Hence the second part of the hypothesis needs further analyses such as evaluation of statistical significance of the difference between the individual ranking lists.

Acknowledgments. This paper was supported by the Project of Technical University of Sofia, Bulgaria, Research and Development Sector (2012): “Classification for Content-based Image Similarity Search for Application in Education and Internet”, 121PD0053-07.

References

1. Seng, W.C., Mirisaee, S.H.: A Content-Based Retrieval System for Blood Cells Images. In: International Conference on Future Computer and Communications (ICFCC 2009), Kuala Lumpur, pp. 412–415 (2009)
2. Depeursinge, A., Muller, H.: Fusion techniques for combining textual and visual information retrieval. In: ImageCLEF: Experimental Evaluation in Visual Information Retrieval. Springer, Heidelberg (2010)

3. Chatzichristofis, S.A., Zagoris, K., Boutalis, Y., Arampatzis, A.: A Fuzzy Rank-Based Late Fusion Method for Image Retrieval. In: Schoeffmann, K., Merialdo, B., Hauptmann, A.G., Ngo, C.-W., Andreopoulos, Y., Breiteneder, C. (eds.) MMM 2012. LNCS, vol. 7131, pp. 463–472. Springer, Heidelberg (2012)
4. Chatzichristofis, S.A., Arampatzis, A., Boutalis, Y.: Investigating the Behavior of Compact Composite Descriptors in Early Fusion, Late Fusion and Distributed Image Retrieval. J. Radioengineering 19(4), 725–733 (2010)
5. Gan, R., Yin, J.: Using LIRe to Implement Image Retrieval System Based on Multi-feature Descriptor. In: ICDMA, GuiLin, pp. 1014–1017 (2012)
6. Dubey, R.S., Choubey, R., Bhattacharga, J.: Multi Feature Content Based Image Retrieval (IJCSE) International Journal on Computer Science and Engineering 2(6), 2145–2149 (2010)
7. Kimura, P.A.S., Cavalcanti, J.M.B., Saraiva, P.C., Torres, R., da, S., Gonçalves, M.A.: Evaluating retrieval effectiveness of descriptors for searching in large image databases. Journal of Information and Data Management 2(3), 305–320 (2011)
8. Hiremath, P.S., Pujari, J.: Content Based Image Retrieval based on Color, Texture and Shape features using Image and its complement. International Journal of Computer Science and Security 1(4), 25–35 (2007)
9. Popova, A.A., Neshov, N.N.: Image Similarity Search Approach Based On The Best Features Ranking. Egyptian Computer Science Journal 37(1), 51–65 (2013)
10. Lux, M., Chatzichristofis, S.: LIRe: Lucene Image Retrieval – An Extensible Java CBIR Library. In: Proceedings of the 16th ACM International Conference on Multimedia, Vancouver, Canada, pp. 1085–1088 (2008)
11. Lux, M.: Content Based Image Retrieval with LIRE. In: Proceedings of the 19th ACM International Conference on Multimedia, Scottsdale, Arizona, USA, pp. 735–738 (2011)
12. Wang, J.Z., Li, J., Wiederhold, G.: SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture LIbraries. The IEEE Trans. on Pattern Analysis and Machine Intelligence 23(9), 947–963 (2001)
13. Schaefer, G., Stich, M.: UCID - An Uncompressed Colour Image Database. In: Proc. SPIE, Storage and Retrieval Methods and Applications for Multimedia 2004, San Jose, USA, pp. 472–480 (2004)
14. Shao, H., Gool, L.V.: Zubud-zurich Buildings Database for Image Based Recognition, Swiss FI of Tech., Tech. report no. 260 (2003)
15. Deselaers, T., Keysers, D., Ney, H.: Features for Image Retrieval: An Experimental Comparison. Information Retrieval 11(2), 77–107 (2008)
16. Vogt, C., Cottrell, G.: Predicting the Performance of Linearly Combined IR Systems. In: 21th Annual Intl ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 190–196. ACM Press, New York (1998)

Vehicle Plate Recognition Using Improved Neocognitron Neural Network

Dmitry Kangin[1], George Kolev[2], and Plamen Angelov[1]

[1] InfoLab21, Lancaster University, Lancaster LA1 4WA United Kingdom
{d.kangin,p.angelov}@lancaster.ac.uk

[2] Center of Coordination of International Scientific Education Programmes, Russia, Moscow
mail@kolev.us

Abstract. This paper describes a novel vehicle plate recognition algorithm based on text detection and improved neocognitron neural network, similar to [1] and based on Fukushima's neocognitron. The proposed recognition algorithm allows us to improve the recognition speed and accuracy comparing to both traditional neocognitron and some state-of-art algorithms (multilayer perceptron, topological methods). It can be used as a solution for image classification and analysis tasks. As an example, the neocognitron can be utilized for symbols recognition [2]. We propose several modifications comparing to the Fukushima's modification of the neocognitron: namely, layer dimensions adjustment, threshold function and connection Gaussian kernel parameters estimation. The patterns' width and height are taken into account independently in order to improve the recognition of patterns of slightly different dimensions. The learning and recognition calculations are performed as FFT convolutions in order to overcome the complexity of the neocognitron output calculations. The algorithm was tested on low-resolution (360×288) video sequences and gave more accurate results comparing to the state-of-the-art methods for low-resolution test set.

Keywords: vehicle plates recognition, image segmentation, Chan-Vese algorithm, neocognitron neural network.

1 Introduction

The vehicle identification is utilized in various applications:

- road traffic control systems;
- vehicle plate identification systems;
- parking control.

Most of state-of-the-art systems use vehicle plate recognition for vehicle identification. Systems used for vehicle plate recognition use one of two different ways of implementation. Identification based on a vehicle photo uses motion detection in order to obtain the object and takes a photo of moving object by

© Springer-Verlag Berlin Heidelberg 2013

high-resolution camera (2-5 MPix), when motion detector triggers. So, we cannot track the vehicle plate in order to obtain the best result (according to some particular criteria) from sequential frames, but we have a higher image resolution. Alternative way is to use video detectors in order to obtain the frame sequence and form the result according several frames.

Typically, the vehicle plate recognition task is composed of several stages: vehicle plates extraction, vehicle plates symbols extraction, and vehicle plates symbols recognition. Various methods were implemented in order to realize each stage of the algorithm.

Vehicle plates extraction can be implemented in different ways:

– based on line detection algorithms ([3]),
– based on template matching ([4]),
– based on text detection ([5]),
– morphological filtering-based candidate region extraction ([6]).

Vehicle plates symbols extraction methods are more wide-spread:

– binarization-based segmentation ([7]);
– based on text detection ([5]);
– watershed segmentation ([8]),
– segmentation based on functional minimization ([9]);
– feature-based segmentation ([10]).

However, this list is not exhaustive as various methods of segmentation were developed during the last decades.

Vehicle plate symbol recognition methods are also widely described as it is a particular case of a wide class of symbol recognition methods, based on neural networks ([1], [2]), decision trees [11], support vector machines [12].

The proposed method is based on the neocognitron neural network that is a convolution neural network using as input data the spatial representation of the image. However, there are various convolution neural networks based on another data, such as Bruna-Mallat wavelet scattering network [13]. Both these network combine unsupervised learning with supervised one. Comparing to this network, we work with spacial image, even using Fourier transform for convolution calculation, as will be shown in the modified neocognitron description.

Some authors use various frequency spaces or curvature space [14]. It gives a possibility for invariance to rotation and translation, but have large issues with distorted patterns due to even slight modifications in distortion can give a great modification in curvature space. The same issue comes when using a topological criteria [15]. For instance, using homology groups, we define an equivalence class by the arbitrary continious transformations of the generating element. So, these classes should measure a distance between these classes regarding geometrical information as well as topological in order to classify patterns robustly, as it is done for persistent homology [16].

The method, proposed in this article, contains several stages:

- text detection plate segmentation and plate symbols extraction;
- symbol text recognition using the neocognitron neural network.

The proposed neocognitron neural network is used for recognition and is similar to the one reported in [17]. It is a slightly modified version compared to the Fukushima's neocognitron neural network [1].

2 Vehicle Plate Segmentation

The line extraction algorithm is widely used in vehicle number plate recognition systems. First, Canny algorithm is used for edge detection [18] (Figure 1). This

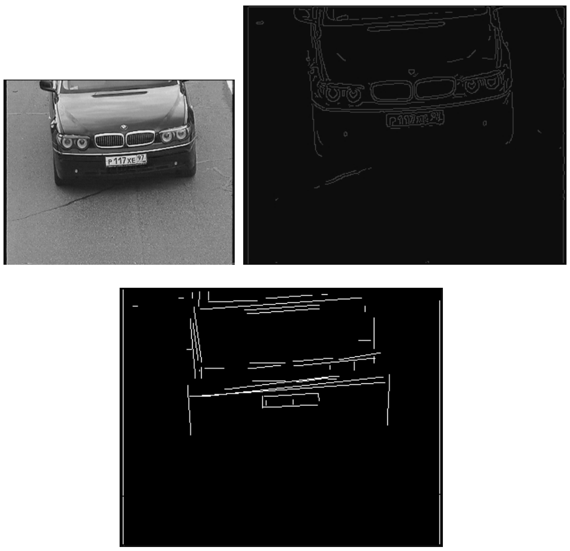


Fig. 1. Canny edge detection and Hough line detection algorithm applied to vehicle images

algorithm provides the edges as the input for Hough transformation algorithm (Figure 1). At the next stage, the bounding rectangle, fitting to the lines, extracted by Hough algorithm in sense of function. The condition of fitting of the rectangle can be written as

$$|LT_y - R_t| < \theta_1, |LB_y - R_b| < \theta_2, |LL_x - R_l| < \theta_3, |LR_x - R_r| < \theta_4. \quad (1)$$

Here θ_i are constants depending on vehicle plate size,
$LT = ((x_{LT1}, y_{LT1}), (x_{LT2}, y_{LT2}))$ — the top line in the rectangle,
$LB = ((x_{LB1}, y_{LB1}), (x_{LB2}, y_{LB2}))$ — the bottom line in the rectangle,

$LL = ((x_{LL1}, y_{LL1}), (x_{LL2}, y_{LL2}))$ — the left line in the rectangle,
$LR = ((x_{LR1}, y_{LR1}), (x_{LR2}, y_{LR2}))$ – the right line of the rectangle,
R_t, R_b, R_l, R_r — the top, bottom, left and right coordinates of the rectangle,
$L\{B, T\}_y = 1/2(Y_{L\{B,T\}1} + Y_{L\{B,T\}2})$,
$L\{L, R\}_x = 1/2(X_{L\{L,R\}1} + X_{L\{L,R\}2})$. However, such algorithm has some principal restrictions (Figure 4):

– it works only for contrasty plate frames (not for white plate with white frame on white car, for example); however, the plate can be readable in such conditions;
– it cannot handle some deformed plates.

3 Text Detection Algorithm

Text detection algorithm [5] can be applied to vehicle plate detection task with some additional conditions over the sizes of such regions. This class of algorithms has some advantages comparing with vehicle plate segmentation based on the line detection algorithms:

– detection based not on heuristic thresholds, but on learning;
– possibility of detection vehicle plates without frames;
– possibility of detection dirty or curved vehicle plates.

However, if the text is corrupted or erased, the algorithm can miss the plate due to using the text criteria instead of border criteria. According to [5], the text extraction is performed in three stages:

– preliminary region detection by Extremal Regions algorithm;
– detection using region classifier (Real AdaBoost [19] used in [5]);
– region grouping in order to obtain chained text regions.

Let us assume the close subset $\Omega \subset \mathbb{R}^2$ and two-dimensional image $I : \Omega \to [0, 1]$. Let us also assume a region $R \subseteq \Omega$ to be a contiguous in sense of adjacency relation $\Lambda \subset \Omega \times \Omega$. In practical cases, we can determine contiguous region R by conditions (4-connected adjacency relation):

$$\Lambda = \{\forall(x, y) \in \Omega : \{((x \pm 1, y), (x, y)), ((x, y \pm 1), (x, y))\} \cap \Omega \times \Omega\} \qquad (2)$$

$$\forall r \in R \exists q : (r, q) \in \Lambda.$$

It can be easily seen, that this relation is symmetric. Extremal regions have to meet the following conditions:

$$\forall p \in R, q \in \partial R : I(Q) > \theta \geqslant C(p), \qquad (3)$$

where θ denotes threshold value. An extremal region at threshold θ is formed as a union of several ERs at threshold $\theta - \delta$ and pixels of values $(\theta - \delta, \theta]$. So, we have an inclusion relation among extremal regions. This approach allows us to obtain the regions for further processing. The computation technique of ER is described in [5]. At the next stage, some descriptors are calculated in order to select ERs containing text data. These features include:

– area of a region is a power of the pixel set, corresponding to the region;
– bounding box is described by top-right and bottom-left corners of the region;
– perimeter p is the length of the boundary ∂R of the region R;
– Euler number ν, determined as a difference between the number of connected components and the holes;
– horizontal crossings of region R is a number of transitions between pixels belonging to the region and not belonging to it for each of the region rows.

The second classifier stage incorporates more complicated parameters, such as

– convex envelope ratio as the ratio of the convex envelope to the area of the region;
– the number of boundary inflexion points determined as the number of changes of boundary convexity;
– hole area ratio is a ratio of count of pixels belonging to holes to the region area.

The detector uses exhaustive search in order to glue up the different symbols into whole number plates, described in [20].

4 Vehicle Plate Symbols Extraction

Vehicle plate symbol extraction can be derived from the text detection algorithm, but it can also be performed separately by different algorithms of segmentation. One of such symbol segmentation methods based on Canny edge detection algorithm [18] (Figure 2).

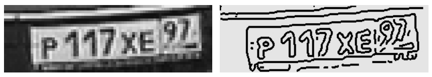


Fig. 2. Vehicle plate edge detection

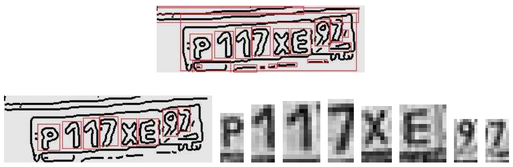


Fig. 3. Symbols envelope determination: envelope determination, non-relevant candidates elimination, symbols segmentation

The method is based on the assumption, that the vehicle plate symbols has a continious outer contour. Then each symbol is expected to have such contour. The size of symbol candidates is thresholded as that the size of the pattern cannot have size below specified value (such situation is regarded as noise) or exceed it. So, the vehicle plate symbols extraction method consists of three stages:

1. Canny edge detection (Figure 2);
2. continious contours rectangular envelope determination (Figure 3);
3. non-relevant candidate elimination using size threshold criteria;

5 Chan-Vese Algorithm for Segmentation

Alternatively, segmentation can be performed using the Chan-Vese algorithm [9]. The method resembles an active contour model [21], in which curve is moved around the object till the stop condition on the boundary of the object occurs, and generalizes Mumford-Shah functional to obtain the rule for production of such boundary curve. According to the Chan-Vese algorithm, the task of image segmentation for object detection is converted to the task of minimization of a characteristic functional. Let us assume the close subset $\Omega \subset \mathbb{R}^2$ and two-dimensional image $I : \Omega \to [0,1]$. Let us also denote a parametric curve $\zeta(s) : [0,1] \to \mathbb{R}^2$, defining a boundary $\partial\omega$ of the open subset $\omega \subset \Omega$. Also some notions are introduced in [9]:

$$\text{inside}(\zeta) = \omega, \text{outside}(\zeta) = \Omega \setminus \omega, \text{length}(\zeta) \text{–the length of } \zeta \text{ curve,}$$

$$\text{area}(\text{inside}(\zeta)) - \text{the area inside the } \zeta \text{ curve.}$$

The Chan-Vese algorithm uses the energy functional

$$\psi(\zeta) = \mu \cdot \text{length}(\zeta) + \nu \cdot \text{area}(\text{inside}(\zeta)) + \tag{4}$$

$$+\lambda_1 \int_{\text{inside}(\zeta)} |I(x,y) - c_1(\zeta)|^2 dxdy + +\lambda_2 \int_{\text{outside}(\zeta)} |I(x,y) - c_2(\zeta)|^2 dxdy.$$

$$\text{Here } c_1(\zeta) = \text{avg}(\text{inside}(\zeta)), c_2(\zeta) = \text{avg}(\text{outside}(\zeta))$$

are functional parameters which depend on ζ curve, μ and ν are coefficients, λ_1 and λ_2 are Lagrangian multipliers. For calculation, the function ϕ is defined as

$$\phi(x,y) = 0, (x,y) \in \zeta, \phi(x,y) > 0, (x,y) \in \text{inside}(\zeta), \phi(x,y) < 0, (x,y) \in \text{outside}(\zeta).$$

Using the Heaviside function, it can be represented for convenience as a numeric representation in the form

$$\psi(\phi) = \mu \int_\omega \delta(\phi(x,y)) |\nabla\phi(x,y)| dxdy + \nu \int_\omega H(\phi(x,y))) dxdy +$$

$$\lambda_1 \int_\omega |I(x,y) - c_1(\phi(x,y))|^2 H(\phi(x,y)) dxdy +$$

$$\lambda_2 \int_\omega |I(x,y) - c_2(\phi(x,y))|^2 [1 - H(\phi(x,y))] dxdy, \tag{5}$$

This task can be regarded as Euler-Lagrange equation and can be solved by finite differences scheme, detailed description of which is given in [9].

6 Modified Neocognitron Neural Network for Vehicle Plates Symbols Recognition

The network contains several layers, connected sequentially:

- input layer U_0;
- contrast extracting layer U_G;
- four groups of stages $U_{S_{1-4}}$, $U_{C_{1-4}}$, each containing sequentially connected S- and C-cells layer.

The input layer U_0 consists of one cell plane and is used for patterns input. The next layer used for contrast extracting cells U_G contains two cell planes: with concentric and off-centre receptive fields. It is used for extraction of the contrast from the images independently of the brightness mean.

$S-$layers contain several cells each and are used for feature extraction, while $C-$layers are used for blurring of the S-layer output.

The first $S-$layer (U_{S_1}) is used for edge component extraction for different rotation angles.

The S-cells' connections at stages U_{S_2} and U_{S_3} are trained by unsupervised competitive learning as described in [1], [2], [22]. However, U_{S_4} layer is trained by supervised competitive learning according to [1] and is used to provide the recognition results.

For all $S-$cells, the resulting output is calculated according to the formulae

$$u_{S_l}(\overrightarrow{n},k) = \frac{\Theta_l}{1-\Theta_l}\phi\lceil w_{S_l}(\overrightarrow{n},k) - 1\rceil, \tag{6}$$

$$w_{S_l}(\overrightarrow{n},k) = \frac{1 + \sum_{\xi=1}^{K_{C_{l-1}}} \sum_{|\nu|<A_{S_l}} a_{S_l} u_{C_{l-1}}(\overrightarrow{n} + \overrightarrow{\nu}, \xi)}{1 + \theta_l b_{S_l}(k) v_l(\overrightarrow{n})}, \tag{7}$$

$$v_l(\overrightarrow{n}) = \sqrt{\sum_{\xi=1}^{K_{C_{l-1}}} \sum_{|\overrightarrow{\nu}|<A_{S_l}} c_{S_l}(\nu)\{u_{C_{l-1}}(\overrightarrow{n} + \overrightarrow{\nu}, \xi)\}^2}, \tag{8}$$

$$b_{S_l}(k) = \sqrt{\sum_{\xi=1}^{K_{C_{l-1}}} \sum_{|\overrightarrow{\nu}|<A_{S_l}} \frac{(a_{S_l})^2}{c_{S_l}(\overrightarrow{\nu}))}}. \tag{9}$$

Here $\phi(x) = \max(x,0)$, $\psi(x) = \phi(x)/(1+\phi(x))$.

Parameter $\overrightarrow{n}$ is a location of the cell plane, while k is the number of the cell of the stage. The θ_l parameter is a threshold, resulting in selectivity of the features. a_{S_l} represents the fixed strength of the connection coming from the preceding stage's C-cells. Its size A_{S_l} represents the summation range for the input connections. $K_{C_{l-1}}$ represents the count of planes for the previous C-layer. $c_{S_l}(\nu)$ is a weight function affecting the v_l output. The dimensions of this functions are equal to a_{S_l} dimensions.

For $C-$cells of layers $U_{C_{1-3}}$ the outputs are given by expressions

$$u_{C_l}(\overrightarrow{n}, k) = \psi\{ \sum_{|\nu|<A_{C_l}} a_{C_l}(\overrightarrow{\nu}) u_{S_l}(\overrightarrow{n} + \overrightarrow{\nu}, k)\}. \tag{10}$$

The layer U_{C_4} has the cells each corresponding to one pattern class. The connections from layer U_{S_3} are transmitting the results to the layer U_{C_4}. For this stage, the $C-$cells are uniting the $S-$cells to the recognition result classes. The competition amongst the $S-$cells are performed, so that no more than one maximal recognition result may be transmitted to the next layer (U_{C_4}). The modified neocognitron that we propose uses FFT [23] for input and connection convolutions in order to provide the usage of standard devices performing FFT.

The calculation of the output values of the layers can be regarded as the convolution of the input values and connections, that is a common approach in different signal processing applications. Such interpretation gives us the ability to use the convolution theorem:

$$\mathfrak{F}(f \otimes h) = \mathfrak{F}(f) \times \mathfrak{F}(h), \tag{11}$$

where operator $\mathfrak{F}$ denotes the discrete Fourier transform (DFT), operator $\otimes$ is convolution, operator $\times$ refers to element-wise multiplication, and f, h — input and connections matrices.

The layers are self-organized by competitive learning. The S-planes count is not predefined, but is generated if no suitable planes are found according to the "winner-takes-all" process.

In order to provide outputs calculation, the discrete convolution theorem is used. However, the convolution is assumed to be cyclic, so that certain restrictions are applied to input and connection data. To exclude impact of the error due to boundary conditions, we set the size of convolution, connection and input matrices to

$$N_{ox} = N_{ix} + N_{cx} - 1, N_{oy} = N_{iy} + N_{cy} - 1, \tag{12}$$

N_{ix}, N_{iy} are dimensions of the required output matrix, N_{cx}, N_{cy} denote dimensions of the connection matrix. Both input and connection matrices are initially filled by zeros.

The next step is the element-wise multiplication of input and connection FFT matrices:

$$\mathfrak{F}(o) = \mathfrak{F}(f) \times \mathfrak{F}(h), o = \mathfrak{F}^{-1}(\mathfrak{F}(o)). \tag{13}$$

Summing up, we can describe the sequence of the convolution calculation consisting of the following steps:

1. input and convolution matrices FFT calculation;
2. element-wise input and FFT multiplication of connection matrices;
3. obtaining the resulting matrix inverse FFT;
4. S-cells thinning-out, performed over the matrix, obtained from the previous step.

7 The Experimental Set Size Estimation

The experimental set size estimation is important for determination of benchmarking accuracy. In this section we describe the initial assumptions on the test set and the model, used for testing data set size, analogously to [24]. We assume to consider as a separate pattern each symbol of the vehicle plate. We assume that the data elements has a probability distribution:

$$P(\text{pattern}, \text{class}) = P(\text{pattern})P(\text{class}|\text{pattern}). \tag{14}$$

We can assume also, that the classes should contain all the variants of symbols. Also we assume the binary events, corresponding to the tuple "recognizer", as "1" for erroneous recognition and "0" for correct recognition. Then we assume that the number of errors is calculated by the binomial distribution:

$$P(k) = C_n^k p^k (1-p)^{n-k}, F(k) = \sum_{i=1}^{k} P(i). \tag{15}$$

where error mean is np and variance is $np(1-p)$. A guaranteed estimator for test set capacity states, that, with the probability $(1-\alpha)$, error probability p will not exceed $\hat{p} = k/n$ more than $\varepsilon(n, a) = \beta p$:

$$\text{Prob}(p >= \hat{p} + \varepsilon(n, \alpha)) = F(n) \geqslant 0. \tag{16}$$

Using the estimation of binomial law by the Normal law, as it is described in [24], we can obtain the test set size estimation as:

$$n = \frac{-2 \ln \alpha}{\beta^2 p}. \tag{17}$$

Assuming $\alpha = 0.05$, $\beta = 0.2$, $p = 0.03$ and a pessimistic bound, we obtain, that the count of test set elements is $n = 5000$. The count of plates for vehicle plate detection and segmentation algorithms estimation were chosen in the same way, so that $m = 5000$ video fragments with vehicle plates depicted were chosen.

The video fragments have the following characteristics: black-and-white (intensity depth 8 bit), frame size 320×288 pixels, frame rate 25 fps. The vehicle plate location and existance on each particular frame is not pre-defined as well as the direction of vehicle motion.

8 Experimental Evaluations

The algorithm results are depicted in Table 1. For the first stage, the vehicle plate detection algorithm, the results depict the ability of the algorithm for proper vehicle detection. As we can see, line detection algorithm have a lot of misses caused by low contrast or deformed of vehicle plate frame. The causes of such low detection rate for this algorithm are shown in Figure 4. Common issue for all of these algorithms is the inability to deal with vehicle plates with screened

Table 1. The results of experimental evaluations

Algorithm	True recognition	Miss	False hit
Vehicle plate detection			
Text detection	0.93	0.02	0.05
Line detection	0.65	0.30	0.05
Symbols segmentation			
Text detection	0.85	0.12	0.03
Chan-Vese algorithm	0.83	0.14	0.03
Edge extraction algorithm	0.75	0.2	0.05
Vehicle plate symbols recognition			
Neocognitron NN	0.96	-	-

letters. However, the algorithm of text detection have the softest restriction as only symbols, but no borders are required for vehicle plate detection. All of the algorithms are restricted in rotation invariance. For this test set, the vehicle plate rotation was varying up to ten degrees.

For the second stage, the vehicle plate detection algorithm, the results show the possibility of the algorithm to determine the positions of each symbols on the correctly extracted plate. Common issue for all of the described algorithms of symbols detection is the inability to deal with glued or partly erased symbols (Figure 5).

For the third stage, the symbol set was chosen as a subset of the results of the previous algorithm. Tests for symbols recognition were performed using one learning and one recognition set, each containing 5000 patterns, i.e. images of various sizes up to 14×17 pixels, which depict the Russian vehicle plate symbols. Although the recognition rate is dependent on the training patterns, the proposed recognition demonstrated up to 96% recognition success rate, whereas the learning set yielded 100% of recognition. The examples of the learning and test pattern lists are shown in Figure 6.

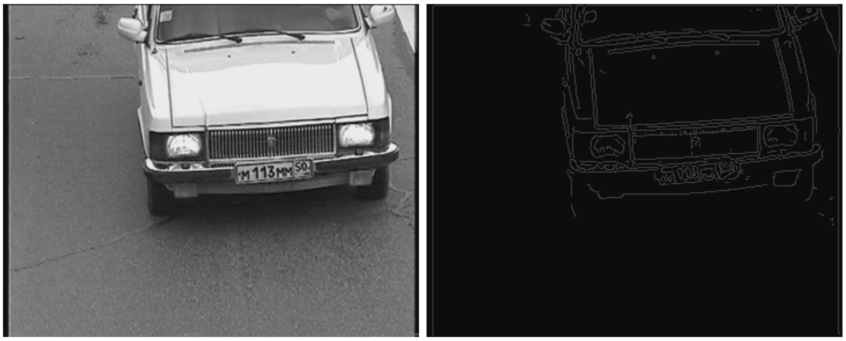


Fig. 4. Canny edge detection over low contrast and deformed vehicle plate

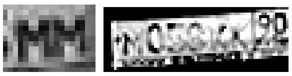


Fig. 5. Glued and partly erased vehicle plate symbols

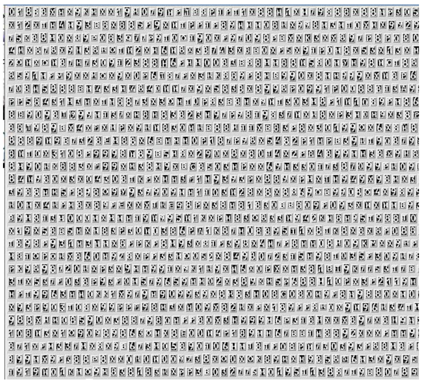


Fig. 6. Learning and test pattern examples

9 Conclusion

In the performed research, the novel method of vehicle plate recognition was presented as well as comparison against existing vehicle plate recognition systems. The method of scene text detection showed a much better results (0.93 against 0.65 for line detecion algorithm) for vehicle plate detection as well as for text symbols segmentation. For symbols recognition, the neocognitron neural network allowed to achieve 96% of correct symbols recognition. The results of this work can be utilized in automatic vehicle plate recognition for parkings and road traffic analysis systems. As the result, the proposed algorithm have achieved better recognition results on low-resolution vehicle image data.

The proposed algorithm does not depend on video sequence data, but takes into account only information on each video frame separately, that gives us a possibility to generalize it for vehicle image data.

References

1. Fukushima, K.: Neocognitron for handwritten digit recognition. Neurocomputing 51, 161–180 (2003)
2. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4), 93–202 (1980)

3. Kamat, V., Ganesan, S.: An Efficient Implementation of the Hough Transform for Detecting Vehicle License Plates Using DSPS. In: Proceedings of Real-Time Technology and Applications, pp. 58–59 (1995)
4. Yohimori, S., Mitsukura, Y., Fukumi, M., Akamatsu, N., Pedrycz, W.: License plate detection system by using threshold function and improved template matching method. In: IEEE Annual Meeting Fuzzy Information Processing NAFIPS 2004, vol. 1, pp. 357–362 (2004)
5. Neumann, L., Matas, J.: Real-Time Scene Text Localization and Recognition. In: CVPR 2012, Providence, Rhode Island, USA (2012)
6. Ashoori-Lalimi, M., Ghofrani, S.: An Efficient Method for Vehicle License Plate Detection in Complex Scenes. Circuits and Systems 2, 320–325 (2011)
7. Bar-Yosef, I., Beckman, I., Kedem, K., Dinstein, I.: Binarization, character extraction, and writer identification of historical Hebrew calligraphy documents. Springer (2007), doi:10.1007/s10032-007-0041-5
8. Pratyusha, Y.S., Murthy, N.S., Sri RamaKrishna, K.: An Efficient Technique for Segmentation of Characters of Vehicle Identification Number Using Watershed Algorithm. International Journal of Advanced Engineering Sciences and Technologies 5(2), 187–194 (2011)
9. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Transactions on Image Processing 10(2) (February 2001)
10. Ying, H., Song, J., Ren, X.: Character segmentation for license plate by the separator symbols frame of reference. In: 2010 International Conference on Information Networking and Automation (ICINA), vol. 1, pp V1-438 - V1-442 (2010)
11. Drucker, H.: Fast Decision Tree Ensembles for Optical Character Recognition. In: Fifth Annual Symposium on Document Analysis and Information Retrieval, April 15-17 (1996)
12. Malon, C., Uchida, S., Suzuki, M.: Support Vector Machines for Mathematical Symbol Recognition. The institute of electronics, information and communication engineers technical report of IEICE (2006)
13. Bruna, J., Mallat, S.: Invariant Scattering Convolution Networks. IEEE Trans. on PAMI (to appear, 2013)
14. Kamvysselis, M.: Wavelet-based character recognition in curvature space. MIT Machine Learning (6.891, Paul Viola) Final Project
15. Wylie, S., Hilton, P.J.: Homology theory. An introduction to algebraic topology. Bull. Amer. Math. Soc. 70(3), 333–335 (1964)
16. Ghrist, R.: Barcodes: The Persistent Topology of Data. Bulletin (New Series) of the American Mathematical Society 45(1), 61–75 (2008) S 0273-0979(07)01191-3
17. Kangin, D., Kolev, G., Vikhoreva, A.: Further Parameters Estimation of Neocognitron Neural Network Modification with FFT Convolution. Journal of Telecommunication Electronic and Computer Engineering 4(2) (July-December 2012)
18. Canny, J.: A Computational Approach To Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence 8(6), 679–698 (1986)
19. Shapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37, 297–336 (1999)
20. Neumann, L., Matas, J.: Text localization in real world images based on conditional random field. In: ICDAR-2011, pp. 687–691 (2011)
21. Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A., Yezzy, A.: Gradient flows and geometric active contour models. In: Proc. Int. Conf. Computer Vision, Cambridge, MA, pp. 810–815 (1995)

22. Kohonen, T.: Self-organizing maps. Springer, Heidelberg (2001)
23. Briggs, W.L., Henson, V.E.: The DFT: an owner's manual for the discrete Fourier transform, pp. 143–179. Society of Industrial and Applied Mathematics, PA (1995)
24. Guyon, I., Makhoul, J., Schwartz, R.M., Vapnik, V.: What Size Test Set Gives Good Error Rate Estimates? IEEE Transactions on Pattern Analysis and Machine Intelligence 20(1), 52–64 (1998)

Author Index

Adams, Rod 319
Aknin, Patrice 9
Alaíz, Carlos M. 66
Alexiev, Kiril 343
Alimi, Adel M. 74, 511
Alonso-Weber, Juan Manuel 335
Alva, Parimala 319
Ammar, Boudour 74, 511
Angelov, Plamen 194, 628
Anguita, Davide 426, 434
Anwar, Muhammad Naveed 375
Austin, James 98
Axenie, Cristian 240

Barbero, Álvaro 66
Barros, Pablo V.A. 412
Bauer-Wersing, Ute 603
Berlemont, Samuel 381
Bezerra, Byron L.D. 412
Birvinskas, Darius 208
Bisneto, Juvenal M.M. 412
Bócsi, Botond Attila 1
Bogdan, Martin 224, 442
Botella-Rocamora, Paloma 451
Boukabou, Abdelkrim 503
Boukens, Mohammed 503
Brahmi, Hajer 74
Brombach, Joris 563
Brosch, Tobias 272
Burles, Nathan 98

Cabessa, Jérémie 58
Castellano, Marta 264
Chérif, Farouk 74, 511
Chernodub, Artem 138
Cho, KyungHyun 106, 611
Chouikhi, Naima 511
Cichocki, Andrzej 90
Clares Tomas, Juan Antonio 42
Conradt, Jörg 240, 248, 467
Csató, Lehel 1, 170
Cvek, Urška 351

Damasevicius, Robertas 208
Davey, Neil 319
De la Torre, Fernando 288
Denk, Christian 467
Dervos, Dimitris A. 34
de Sousa, Giseli 319
Dorronsoro, José R. 66

Eggert, Julian P. 367
El Assaad, Hani 9
Evangelidis, Georgios 34

Farkaš, Igor 154
Felder, Martin 563
Fernandes, Bruno J.T. 412
Fernandes, Sérgio M.M. 412
Förger, Klaus 312
Frandina, Salvatore 82, 186
Fujii, Robert H. 296
Furber, Steve 467

Galluppi, Francesco 467
Garcia, Christophe 381, 397
Gardner, Brian 256
Georgieva, Petia 288
Ghio, Alessandro 426, 434
Gnecco, Giorgio 146
Gorce, Philippe 511
Gori, Marco 82, 146
Govaert, Gérard 9
Gross, Horst-Michael 367
Grüning, André 256
Guijarro-Berdiñas, Bertha 122
Gutierrez, German 335

Hagiwara, Masafumi 595
Haschke, Robert 178
Hasler, Stephan 603
Hatori, Yasuhiro 327
Heinrich, Stefan 216
Hercus, Robert 527
Higuchi, Daisuke 162
Ho, Kim-Fong 527
Hocke, Jens 420
Homma, Yukinori 595
Honkela, Timo 312, 405, 579

Idrissi, Khalid 397
Ilin, Alexander 106
Irifune, Mayumi 296
Ishiguro, Hiroshi 17
Ishii, Shin 17
Ivliev, Sergey 194

Jakab, Hunor Sandor 170
Jeanson, Francis 280
Jusas, Vacius 208

Kaifel, Anton 563
Kanellos, Fotis D. 586
Kangin, Dmitry 628
Karamitopoulos, Leonidas 34
Karhunen, Juha 106
Kasneci, Gjergji 442
Khosravi, Abbas 389
Kilgore, Phillip C.S.R. 351
Kmet, Tibor 483
Kmetova, Maria 483
Knapek, Michael 579
Kolev, Denis 194
Kolev, George 628
Koprinkova-Hristova, Petia 343
Koprinska, Irena 389, 571
Kryzhanovsky, Vladimir 42
Kübler, Thomas C. 442

Lagus, Krista 405
Lang, Bo 359
Layher, Georg 232
Ledezma, Agapito 335
Lefebvre, Grégoire 381
Lippi, Marco 82, 186
Llobet-Blandino, Francisco 467
Lomp, Oliver 475

Maex, Reinoud 319
Maggini, Marco 82, 186
Mamalet, Franck 381
Markarian, Garegin 194
Martel, Julien 397
Martinetz, Thomas 420
Martisius, Ignas 208
Mashita, Tatsuroh 327
Mastorakis, Nikos E. 586
Meier, Martin 178
Meis, Jon 563
Melacci, Stefano 82, 146, 186
Mladenov, Valeri 586
Moreno Arostegui, Juan Manuel 519

Nakamura, Yutaka 17
Nakashika, Toru 397
Neshov, Nikolay N. 619
Neumann, Heiko 232, 272
Nieminen, Ilari T. 405

Oakes, Michael 375
Okadome, Yuya 17
Okamoto, Hiroshi 50
O'Keefe, Simon 98
Olivier, Paul 519
Oneto, Luca 426, 434
Osana, Yuko 495
Ougiaroglou, Stefanos 34
Ozawa, Seiichi 162

Panchev, Christo 375
Pardo, Juan 451
Parra, Xavier 426
Peteiro-Barral, Diego 122
Petrov, Michail 459
Phan, Anh-Huy 90
Pipa, Gordon 26, 264
Plana, Luis A. 467

Raiko, Tapani 106
Rana, Mashud 389, 571
Rañó, Iñaki 475
Rao, Junyang 545
Rebrová, Kristína 154
Reyes-Ortiz, Jorge Luis 426, 434
Rezzoug, Nasser 511
Richter, Mathis 475
Ridella, Sandro 434
Ritter, Helge J. 178
Romeu, Pablo 451
Rosenstiel, Wolfgang 224, 442

Saari, Juho 405
Saitoh, Fumiaki 537
Sakai, Ko 327
Samé, Allou 9
Sanchis, Araceli 335
Sandamirskaya, Yulia 248
Sanguineti, Marcello 146

Schöner, Gregor 475
Schumacher, Johannes 26
Schwabe, Lars 304
Schwenker, Friedhelm 272
Sehnke, Frank 130, 563
Sesmero, M. Paz 335
Shikauchi, Yumi 17
Šíma, Jiří 114
Spüler, Martin 224
Steuber, Volker 319
Strunk, Achim 563
Struwe, Marvin 603
Suvorov, Mikhail 194
Szupiluk, Ryszard 553

Tafaj, Enkelejda 442
Takamatsu, Tsubasa 495
Tatoglou, Christos 34
Terzyiska, Margarita 459
T.M. Júnior, Nestor 412
Todorov, Yancho 459
Torben-Nielsen, Ben 319
Toutounji, Hazem 26
Trutschl, Marjan 351
Tschechne, Stephan 232
Tsekouras, George J. 586

Utani, Akihide 537

Villa, Alessandro E.P. 58
Vollmer, Christian 367

Wang, Tinghua 545
Weber, Cornelius 216
Wei, Hui 359
Wermter, Stefan 216
White, Tony 280
Wojewnik, Piotr 553
Wong, Kit-Yee 527

Ząbkowski, Tomasz 553
Zamora-Martínez, Francisco 451
Zdunek, Rafał 90
Zhao, Dongyan 545
Zhelavskaya, Irina 42
Zheng, Youwei 304
Zibner, Stephan Klaus Ulrich 475
Zuo, Qingsong 359
Zvikhachevskiy, Dmitry 194