

LNCS 6354

Konstantinos Diamantaras
Wlodek Duch
Lazaros S. Iliadis (Eds.)

Artificial Neural Networks – ICANN 2010

20th International Conference
Thessaloniki, Greece, September 2010
Proceedings, Part III

3
Part III

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

Konstantinos Diamantaras Wlodek Duch
Lazaros S. Iliadis (Eds.)

Artificial Neural Networks – ICANN 2010

20th International Conference
Thessaloniki, Greece, September 15-18, 2010
Proceedings, Part III



Springer

Volume Editors

Konstantinos Diamantaras
TEI of Thessaloniki, Department of Informatics
57400 Sindos, Greece
E-mail: kdiamant@it.teithe.gr

Wlodek Duch
Nicolaus Copernicus University
School of Physics, Astronomy, and Informatics
Department of Informatics
ul. Grudziadzka 5, 87-100 Torun, Poland
E-mail: duch@phys.uni.torun.pl

Lazaros S. Iliadis
Democritus University of Thrace, Department of Forestry
and Management of the Environment and Natural Resources
Pantazidou 193, 68200 Orestiada Thrace, Greece
E-mail: liliadis@fmenr.duth.gr

Library of Congress Control Number: 2010933964

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

ISSN 0302-9743
ISBN-10 3-642-15824-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-15824-7 Springer Berlin Heidelberg New York

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

springer.com

© Springer-Verlag Berlin Heidelberg 2010
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

This volume is part of the three-volume proceedings of the 20th International Conference on Artificial Neural Networks (ICANN 2010) that was held in Thessaloniki, Greece during September 15–18, 2010.

ICANN is an annual meeting sponsored by the European Neural Network Society (ENNS) in cooperation with the International Neural Network Society (INNS) and the Japanese Neural Network Society (JNNS). This series of conferences has been held annually since 1991 in Europe, covering the field of neurocomputing, learning systems and other related areas.

As in the past 19 events, ICANN 2010 provided a distinguished, lively and interdisciplinary discussion forum for researchers and scientists from around the globe. It offered a good chance to discuss the latest advances of research and also all the developments and applications in the area of Artificial Neural Networks (ANNs). ANNs provide an information processing structure inspired by biological nervous systems and they consist of a large number of highly interconnected processing elements (neurons). Each neuron is a simple processor with a limited computing capacity typically restricted to a rule for combining input signals (utilizing an activation function) in order to calculate the output one. Output signals may be sent to other units along connections known as weights that excite or inhibit the signal being communicated. ANNs have the ability “to learn” by example (a large volume of cases) through several iterations without requiring a priori fixed knowledge of the relationships between process parameters.

The rapid evolution of ANNs during the last decades has resulted in their expansion in various diverse scientific fields, like engineering, computer science, mathematics, artificial intelligence, biology, environmental science, operations research and neuroscience. ANNs perform tasks like pattern recognition, image and signal processing, control, classification and many others.

In 2010 ICANN was organized by the following institutions: Aristotle University of Thessaloniki, University of Macedonia at Thessaloniki, Technological Educational Institute of Thessaloniki, Hellenic International University and Democritus University of Thrace.

The conference was held in the Kapsis Hotel and conference center in Thessaloniki, Greece. The participants were able to enjoy the atmosphere and the cultural heritage of Thessaloniki, which is built by the seaside and has a glorious history of 2300 years.

As a matter of fact, a total of 241 research papers were submitted to the conference for consideration. All of the submissions were peer reviewed by at least two academic referees. The international Program Committee of ICANN 2010 carefully selected 102 submissions (42%) to be accepted as full papers. Additionally 68 papers were selected for short presentation and 29 as posters.

The full papers have up to 10 pages, short ones have up to 6 pages and posters have up to 4 pages in the proceedings.

In addition to the regular papers, the technical program featured four keynote plenary lectures by the following worldwide renowned scholars:

- Prof. Alessandro E.P. Villa: NeuroHeuristic Research Group, Information Science Institute, University of Lausanne, Switzerland and Institut des Neurosciences, Université Joseph Fourier, Grenoble, France. Subject: “Spatiotemporal Firing Patterns and Dynamical Systems in Neural Networks”;
- Prof. Stephen Grossberg: Department of Cognitive and Neural Systems, Center for Adaptive Systems, and Center of Excellence for Learning in Education, Science, and Technology, Boston University. Subject: “The Predictive Brain: Autonomous Search, Learning, Recognition, and Navigation in a Changing World”;
- Prof. Sergios Theodoridis: Department of Informatics and Telecommunications, National and Kapodistrian University of Athens. Subject: “Adaptive Learning in a World of Projections”;
- Prof. Nikola Kasabov: Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology. Subject: “Evolving Integrative Spiking Neural Networks: A Computational Intelligence Approach”.

Also two tutorials were organized on the following topics:

- Prof. J.G. Taylor: Department of Mathematics, King’s College London. Subject: “Attention versus Consciousness: Independent or Conjoined?”;
- Dr. Kostas Karpuzis: Image, Video and Multimedia Systems Lab, Institute of Communication and Computer Systems (ICCS/NTUA). Subject: “User Modelling and Machine Learning for Affective and Assistive Computing”.

Finally three workshops were organized namely:

- The First Consciousness Versus Attention Workshop (CVA);
- The Intelligent Environmental Monitoring, Modelling and Management Systems for Better QoL Workshop (IEM3);
- The First Self-Organizing Incremental Neural Network Workshop (SOINN).

The ENNS offered 12 travel grants to students who participated actively in the conference by presenting a research paper, and a competition was held between students for the best paper award.

The three-volume proceedings contain research papers covering the following topics: adaptive algorithms and systems, ANN applications, Bayesian ANNs, bio inspired-spiking ANNs, biomedical ANNs, data analysis and pattern recognition, clustering, computational intelligence, computational neuroscience, cryptography algorithms, feature selection/parameter identification and dimensionality reduction, filtering, genetic-evolutionary algorithms, image, video and audio processing, kernel algorithms and support vector machines, learning algorithms and systems, natural language processing, optimization, recurrent ANNs, reinforcement learning, robotics, and self organizing ANNs.

As General Co-chairs and PC Co-chair and in the name of all members of the Steering Committee, we would like to thank all the keynote invited speakers and the tutorial-workshops' organizers as well. Also, thanks are due to all the reviewers and the authors of submitted papers. Moreover, we would like to thank the members of the Organizing Committee headed by Prof. Yannis Manolopoulos and Prof. Ioannis Vlahavas. In particular, we wish to thank Dr. Maria Kontaki for her assistance and support towards the organization of this conference.

Additionally, we would like to thank the members of the Board of the European Neural Network Society for entrusting us with the organization of the conference as well as for their assistance. We wish to give our special thanks to Prof. Włodzisław Duch, President of the ENNS, for his invaluable guidance and help all the way.

Finally, we would like to thank Springer for their cooperation in publishing the proceedings in the prestigious series of Lecture Notes in Computer Science. We hope that all of the attendees enjoyed ICANN 2010 and also the conference site in Thessaloniki, both scientifically and socially. We expect that the ideas that have emerged here will result in the production of further innovations for the benefit of science and society.

September 2010

Włodzisław Duch
Kostandinos Diamandaras
Lazaros Iliadis

Organization

Executive Committee

General Chairs	Konstantinos Diamantaras (Alexander TEI of Thessaloniki)
Program Chair	Wlodek Duch (Nikolaus Copernicus University, Torun)
Workshop Chairs	Lazaros Iliadis (Democritus University of Thrace)
Organizing Chairs	Nikola Kasabov (Auckland University of Technology) Kostas Goulianas (Alexander TEI of Thessaloniki) Yannis Manolopoulos (Aristotle University) Ioannis Vlahavas (Aristotle University)
Members	Maria Kontaki (Aristotle University) Alexis Papadimitriou (Aristotle University) Stavros Stavroulakis (Aristotle University)

Referees

Luis Alexandre	T. Glezakos
Cesare Alippi	Giorgio Gnecco
Plamen Angelov	G. Gravanis
Bruno Apolloni	Barbara Hammer
Amir Atiya	Ioannis Hatzilygeroudis
Monica Bianchini	Tom Heskes
Dominic Palmer Brown	Timo Honkela
Ivo Bukovsky	Amir Hussain
F.F. Cai	Sylvain Jaume
Gustavo Camps-Valls	Yaochu Jin
Ke Chen	D. Kalles
Theo Damoulas	Achilles Kameas
Tharam Dillon	Hassan Kazemian
Christina Draganova	Stefanos Kollias
Gerard Dreyfus	D. Kosmopoulos
Peter Erdi	Costas Kotropoulos
Deniz Erdogmus	Jan Koutnik
Pablo Estevez	Konstantinos Koutroumbas
Mauro Gaggero	Vera Kurkova
Christophe Garcia	Diego Liberati
Erol Gelenbe	Aristidis Likas
Christos Georgiadis	I. Maglogiannis
Mark Girolami	Danilo Mandic

Francesco Marcelloni	Athanassios Skodras
Konstantinos Margaritis	S. Spartalis
Thomas Martinetz	Alessandro Sperduti
Matteo Matteucci	Soundararajan Srinivasan
Ali Minai	Andreas Stafylopatis
Nikolaos Mitianoudis	Johan Suykens
Roman Neruda	Johannes Sveinsson
Erkki Oja	Anastasios Tefas
Mihaela Oprea	Athanasios Tsadiras
Karim Ouazzane	Ioannis Tsamardinos
Theofilos Papadimitriou	T. Tsiligkiris
Charis Papadopoulos	Marc Van Hulle
Constantinos Pattichis	Marley Vellasco
Barak Pearlmutter	Michel Verleysen
Elias Pimenidis	Vassilios Verykios
Vincenzo Piuri	Alessandro E.P. Villa
Mark Plumbley	Jun Wang
Manuel Roveri	Aaron Weifeng
Leszek Rutkowski	Yong Xue
Marcello Sanguineti	K. Yialouris
Mike Schuster	Hujun Yin
Hiroshi Shimodaira	Xiaodong Zhang
A. Sideridis	Rodolfo Zunino
Olli Simula	

Sponsoring Institutions

European Neural Network Society (ENNS)
Aristotle University of Thessaloniki
Alexander TEI of Thessaloniki
University of Macedonia
Democritus University of Thrace
International Hellenic University

Table of Contents – Part III

Classification – Pattern Recognition

Deep Bottleneck Classifiers in Supervised Dimension Reduction	1
<i>Elina Parviainen</i>	
Local Modeling Classifier for Microarray Gene-Expression Data	11
<i>Iago Porto-Díaz, Verónica Bolón-Canedo, Amparo Alonso-Betanzos, and Óscar Fontenla-Romero</i>	
Learning of Lateral Connections for Representational Invariant Recognition	21
<i>Christian Keck and Jörg Lücke</i>	
Computational Properties of Probabilistic Neural Networks	31
<i>Jiří Grim and Jan Hora</i>	
Local Minima of a Quadratic Binary Functional with a Quasi-Hebbian Connection Matrix	41
<i>Yakov Karandashev, Boris Kryzhanovsky, and Leonid Litinskii</i>	
A Learned Saliency Predictor for Dynamic Natural Scenes	52
<i>Eleonora Vig, Michael Dorr, Thomas Martinetz, and Erhardt Barth</i>	
Learning a Combination of Heterogeneous Dissimilarities from Incomplete Knowledge	62
<i>Manuel Martín-Merino</i>	
A Bilinear Model for Consistent Topographic Representations	72
<i>Urs Bergmann and Christoph von der Malsburg</i>	
Accelerating Large-Scale Convolutional Neural Networks with Parallel Graphics Multiprocessors	82
<i>Dominik Scherer, Hannes Schulz, and Sven Behnke</i>	
Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition	92
<i>Dominik Scherer, Andreas Müller, and Sven Behnke</i>	
Visual Shape Recognition Neural Network Using BESOM Model	102
<i>Hiroaki Hasegawa and Masafumi Hagiwara</i>	

Comparing Feature Extraction Techniques and Classifiers in the Handwritten Letters Classification Problem	106
<i>Antonio García-Manso, Carlos J. García-Orellana, Horacio M. González-Velasco, Miguel Macías-Macías, and Ramón Gallardo-Caballero</i>	
The Parameter Optimization of the Pulse Coupled Neural Network for the Pattern Recognition	110
<i>Masato Yonekawa and Hiroaki Kurokawa</i>	
The Use of Feed Forward Neural Network for Recognizing Characters of Dactyl Alphabet	114
<i>Roman Zálušký, Emil Raschman, Mário Krajmer, and Daniela Ďuračková</i>	
Detecting DDoS Attack towards DNS Server Using a Neural Network Classifier	118
<i>Jun Wu, Xin Wang, Xiaodong Lee, and Baoping Yan</i>	
Classification Based on Multiple-Resolution Data View	124
<i>Mateusz Kobos and Jacek Mańdziuk</i>	
Identification of the Head-and-Shoulders Technical Analysis Pattern with Neural Networks	130
<i>Achilleas Zapranis and Prodromos Tsinaslanidis</i>	
Analyzing Classification Methods in Multi-label Tasks	137
<i>Araken M. Santos, Laura E.A. Santana, and Anne M. Canuto</i>	
Learning Bimanual Coordination Patterns for Rhythmic Movements....	143
<i>Rikke Amilde Løvlid and Pinar Öztürk</i>	
Classification of Voice Aging Using Parameters Extracted from the Glottal Signal	149
<i>Leonardo Alfredo Forero Mendoza, Edson Cataldo, Marley Vellasco, and Marco Silva</i>	
Learning Algorithms and Systems	
TopoART: A Topology Learning Hierarchical ART Network	157
<i>Marko Tscherepanow</i>	
Policy Gradients for Cryptanalysis	168
<i>Frank Sehnke, Christian Osendorfer, Jan Sölter, Jürgen Schmidhuber, and Ulrich Röhrmair</i>	
Linear Projection Method Based on Information Theoretic Learning....	178
<i>Pablo A. Vera, Pablo A. Estévez, and Jose C. Principe</i>	

Continuous Visual Codebooks with a Limited Branching Tree Growing Neural Gas	188
<i>Marco Kortkamp and Sven Wachsmuth</i>	
An Efficient Collaborative Recommender System Based on k -Separability	198
<i>Georgios Alexandridis, Georgios Siolas, and Andreas Stafylopatis</i>	
Empirical Analysis of the Divergence of Gibbs Sampling Based Learning Algorithms for Restricted Boltzmann Machines	208
<i>Asja Fischer and Christian Igel</i>	
Multitask Semi-supervised Learning with Constraints and Constraint Exceptions	218
<i>Marco Maggini and Tiziano Papini</i>	
Tumble Tree – Reducing Complexity of the Growing Cells Approach	228
<i>Hendrik Annuth and Christian-A. Bohn</i>	
Sentence Extraction by Graph Neural Networks	237
<i>Donatella Muratore, Markus Hagenbuchner, Franco Scarselli, and Ah Chung Tsoi</i>	
Autonomous Generation of Internal Representations for Associative Learning	247
<i>Michaël Garcia Ortiz, Benjamin Dittes, Jannik Fritsch, and Alexander Gepperth</i>	
Improving Accuracy of LVQ Algorithm by Instance Weighting.	257
<i>Marcin Blachnik and Włodzisław Duch</i>	
Multifactor Expectation Maximization for Factor Graphs	267
<i>Jason T. Rolfe and Matthew Cook</i>	
Weighted Learning Vector Quantization to Cost-Sensitive Learning	277
<i>Ning Chen, Bernardete Ribeiro, Armando Vieira, João Duarte, and João Neves</i>	
Solution Space of Perceptron	282
<i>Jau-Chi Huang and Cheng-Yuan Liou</i>	
Natural Language Processing Neural Network for Recall and Inference	286
<i>Tsukasa Sagara and Masafumi Hagiwara</i>	
Nominally Conditioned Linear Regression	290
<i>Yusuke Tanahashi, Ryohei Nakano, and Kazumi Saito</i>	
A Novel Continuous Dual Mode Neural Network in Stereo-Matching Process	294
<i>Lukasz Laskowski</i>	

Learning Invariant Visual Shape Representations from Physics	298
<i>Mathias Franzius and Heiko Wersing</i>	
Algorithms Creating Algorithms	303
<i>Boyan Lalov</i>	
Assessing Statistical Reliability of LiNGAM via Multiscale Bootstrap	309
<i>Yusuke Komatsu, Shohei Shimizu, and Hidetoshi Shimodaira</i>	
Learning with Convex Constraints	315
<i>Marco Gori and Stefano Melacci</i>	
Theoretical Analysis of Cross-Validation(CV)-EM Algorithm	321
<i>Takashi Takenouchi and Kazushi Ikeda</i>	
Application of BSP-Based Computational Cost Model to Predict Parallelization Efficiency of MLP Training Algorithm	327
<i>Volodymyr Turchenko and Lucio Grandinetti</i>	
Dynamic Shape Learning and Forgetting	333
<i>Nikolaos Tsapanos, Anastasios Tefas, and Ioannis Pitas</i>	
On-Line Ensemble-Teacher Learning through a Perceptron Rule with a Margin	339
<i>Kazuyuki Hara, Katsuya Ono, and Seiji Miyoshi</i>	
Model of the Hippocampal Learning of Spatio-temporal Sequences	345
<i>Julien Hirel, Philippe Gaussier, and Mathias Quoy</i>	
Adding Nonlinear System Dynamics to Levenberg-Marquardt Algorithm for Neural Network Control	352
<i>Mikel Larrea, Eloy Iglesias, and Vicente Gómez</i>	

Computational Intelligence

Some Comparisons of Model Complexity in Linear and Neural-Network Approximation	358
<i>Giorgio Gnecco, Věra Kůrková, and Marcello Sanguineti</i>	
A Funny Proverb Generation System Based on <i>Sukashi</i>	368
<i>Hiroaki Yamane and Masafumi Hagiwara</i>	
Supervised Neural Fuzzy Schemes in Video Transmission over Bluetooth	378
<i>Hassan B. Kazemian, Guillaume F. Remy, and Phil Picton</i>	
A Graph Based Framework for Clustering and Characterization of SOM	387
<i>Rakia Jaziri, Khalid Benabdeslem, and Haytham Elghazel</i>	

Clustering Using Elements of Information Theory	397
<i>Daniel de Araújo, Adrião Dória Neto, Jorge Melo, and Allan Martins</i>	
A Path Planning Method for Human Tracking Agents Using Variable-Term Prediction	407
<i>Noriko Takemura, Yutaka Nakamura, Yoshio Matsumoto, and Hiroshi Ishiguro</i>	
Three-Layer Feedforward Structures Smoothly Approximating Polynomial Functions	411
<i>Yuichi Nakamura and Masahiro Nakagawa</i>	
Neural Networks Training for Weapon Selection in First-Person Shooter Games	417
<i>Stelios Petrakis and Anastasios Tefas</i>	
Efficient Confidence Bounds for RBF Networks for Sparse and High Dimensional Data	423
<i>Abner Rodrigues Neto, Mauro Roisenberg, and Guenther Schwedersky Neto</i>	
Large Scale Problem Solving with Neural Networks: The Netflix Prize Case	429
<i>Nicholas Ampazis</i>	
A Cooperative and Penalized Competitive Learning Approach to Gaussian Mixture Clustering	435
<i>Yiu-ming Cheung and Hong Jia</i>	
A Inference Mechanism for Polymer Processing using Rough-Neuro Fuzzy Network	441
<i>Carlos Affonso and Renato Jose Sassi</i>	
IEM3 Workshop	
Data Mining Methods for Quality Assurance in an Environmental Monitoring Network	451
<i>Ioannis N. Athanasiadis, Andrea-Emilio Rizzoli, and Daniel W. Beard</i>	
Predicting QoL Parameters for the Atmospheric Environment in Athens, Greece	457
<i>Ioannis Kyriakidis, Kostas Karatzas, and George Papadourakis</i>	
Investigating Pollen Data with the Aid of Fuzzy Methods	464
<i>Nikolaos Mitrikis, Kostas Karatzas, and Siegfried Jaeger</i>	

A New Neural Model for Traffic Simulation	471
<i>Enrique Mérida Casermeiro, Gabriel Aguilera Venegas, José Luis Galán García, and Pedro Rodríguez Cielos</i>	
System Architecture for a Smart University Building	477
<i>Thanos G. Stavropoulos, Ageliki Tsoliaridou, George Koutitas, Dimitris Vrakas, and Ioannis Vlahavas</i>	
Monitoring and Assessment of Environmental Impact by Persistent Organic Pollutants	483
<i>Jaroslav Urbánek, Karel Brabec, Ladislav Dušek, Ivan Holoubek, Jiří Hřebíček, and Miroslav Kubásek</i>	
A Feature Selection Method for Air Quality Forecasting	489
<i>Luca Mesin, Fiammetta Orione, Riccardo Taormina, and Eros Pasero</i>	

CVA Workshop

How Do Attention, Intention, and Consciousness Interact?	495
<i>Stephen Grossberg</i>	
Consciousness versus Attention	496
<i>John G. Taylor</i>	
On the Fringe of Awareness: The Glance-Look Model of Attention-Emotion Interactions	504
<i>Li Su, Philip Barnard, and Howard Bowman</i>	
No Stopping and No Slowing: Removing Visual Attention with No Effect on Reversals of Phenomenal Appearance.....	510
<i>Alexander Pastukhov, Victoria Vonau, and Jochen Braun</i>	
Modelling Neurotic Psychopathology: Memory, Attention and Symbolization	516
<i>Roseli S. Wedemann, Luís Alfredo V. de Carvalho, and Raul Donangelo</i>	

SOINN Workshop

How to Use the SOINN Software: User's Guide (Version 1.0)	521
<i>Kazuhiro Yamasaki, Naoya Makibuchi, Furao Shen, and Osamu Hasegawa</i>	
Unguided Robot Navigation with Using Continuous Action Space	528
<i>Sirinart Tangruamsub, Manabu Tsuboyama, Aram Kawewong, and Osamu Hasegawa</i>	

Self-Organizing Incremental Neural Network and Its Application	535
<i>Furao Shen and Osamu Hasegawa</i>	
Machine Learning Approaches for Time-Series Data Based on Self-Organizing Incremental Neural Network	541
<i>Shogo Okada, Osamu Hasegawa, and Toyoaki Nishida</i>	
Online Knowledge Acquisition and General Problem Solving in a Real World by Humanoid Robots	551
<i>Naoya Makibuchi, Furao Shen, and Osamu Hasegawa</i>	
Incremental Learning Using Self-Organizing Neural Grove	557
<i>Hirotaka Inoue</i>	
Fast and Incremental Attribute Transferring and Classifying System for Detecting Unseen Object Classes	563
<i>Aram Kawewong and Osamu Hasegawa</i>	
Author Index	569

Deep Bottleneck Classifiers in Supervised Dimension Reduction

Elina Parviainen

B ECS (Dept. of Biomedical Engineering and Computational Science)
Aalto University School of Science and Technology, Finland

Abstract. Deep autoencoder networks have successfully been applied in unsupervised dimension reduction. The autoencoder has a "bottleneck" middle layer of only a few hidden units, which gives a low dimensional representation for the data when the full network is trained to minimize reconstruction error. We propose using a deep bottlenecked neural network in supervised dimension reduction. Instead of trying to reproduce the data, the network is trained to perform classification. Pretraining with restricted Boltzmann machines is combined with supervised finetuning. Finetuning with supervised cost functions has been done, but with cost functions that scale quadratically. Training a bottleneck classifier scales linearly, but still gives results comparable to or sometimes better than two earlier supervised methods.

1 Introduction

This work contributes to the line of research started by [1], where the authors present a way of training a deep neural network by pretraining using restricted Boltzmann machines (RBM), and finetuning the result by backpropagation. Backpropagation, if started from a random initialization, easily ends up in a local optimum, especially in a high-dimensional parameter space like that of a deep network. Pretraining mitigates this problem by finding a good initialization.

In an autoencoder, the finetuning phase minimizes the reconstruction error. By using a different cost function the network can be made to learn a different mapping, for example one that gives results similar to an existing dimension reduction method [2][3]. Also information about class labels can be introduced into the cost function.

To the best of our knowledge, supervised finetuning of deep RBM-pretrained networks has only been done using cost functions based on the concept of neighborhood. The problem of neighborhood-based cost is the necessity to refer to neighboring points when computing the cost for a point, which results in complex cost functions. Two such cost functions, neighborhood components analysis (NCA) [4] and a large margin k-nearest neighbor (kNN) classification cost [5] will be described in detail later.

In this work, we suggest a much faster way of achieving supervision: the network is finetuned to perform nonlinear classification. We add a two-layer classifier after the bottleneck layer, and backpropagate its classification error through the whole network to finetune the weights. Training a classifier network scales linearly in the number of data points.

We start by briefly reviewing earlier work on supervision in bottleneck architectures in Sect. 2 and introduce the bottleneck classifier in Sect. 3. Methods used for comparison are described in Sect. 4. In Sect. 5 and 6 we report our experiments and their results, and conclusions are drawn in Sect. 7.

2 Related Work

Dimension reduction with autoencoders was suggested already in the 1980s, e.g. [6], but training a deep autoencoder remained a challenge until recently. Currently, at least two different strategies for training deep networks are known [7]: training each layer as an RBM or as a shallow autoencoder.

We will compare the bottleneck classifier to two supervised cost functions, which have been used to finetune an RBM-pretrained deep network. Neighborhood components analysis [8] attempts to maximize the expected number of correctly classified points in nearest neighbor classification. The same cost function is used with a deep network in [4], yielding a nonlinear extension of NCA. Another neighborhood-based cost function, for achieving presentations with a large margin in k-nearest neighbor (kNN) classification, is suggested in [5]. It is based on distances to a number of same-class and other-class points.

In addition to these fully supervised cost functions, different combinations of supervised and unsupervised structures have been suggested. A strategy called divergent autoencoding [9] uses a model with a joint bottleneck layer but separate decoder networks for each class. Each decoder is trained with samples from a certain class. Regularizers, either for the network output or each layer in turn, are used for adding supervisory information to a deep network in [10]. Supervision in the form of linear classifiers has been suggested in [11] for a deep architecture with autoencoder pretraining for layers. Each layer is coupled with a linear classifier, and a tuning parameter determines the importance of the two in training. The authors achieve good precision and recall on three document data sets, which makes us expect good results also with the RBM-based architecture combined with classification.

A shallow classifier with a bottleneck layer was used in [12]. The authors studied whether adding class labels would help the network produce presentations which ignore features irrelevant to classification. They used two artificial data sets, small by today's standards, and considered a bottleneck network with three hidden layers. In that setting, they found that the bottleneck classifier improved results over autoencoders. Our work brings this idea up to date, using a deep architecture, modern pretraining and real data sets of tens of thousands of samples.

3 Bottleneck Classifier vs. Other Network Layouts

3.1 RBM Pretraining for a Deep Network

Structure of a restricted Boltzmann machine [13] is a fully connected bipartite graph. Nodes in its visible layer are associated with input data, and those in the hidden layer form a representation of the data. Training tries to find the weights so that the marginal distribution of the hidden nodes would match that of the visible nodes as well as possible. Minimizing the Kullback-Leibler divergence between the two distributions would achieve this, but the gradient contains terms which cannot be computed analytically. The work in [4] goes around this problem by minimizing a slightly different cost function called contrastive divergence. It can be optimized by standard gradient descent.

Pretraining a deep network needs a stack of RBMs, which form the layers of the deep network. They are trained one at a time, using the data as input in the first layer, and training each subsequent layer to reproduce the result of the previous layer as accurately as possible. Sizes of the layers are, of course, determined by the desired layout of the deep network. The weights for encoder part of the network, that is, all layers up to and including the bottleneck layer, are the weights of the RBM stack. For the decoder part (layers after the bottleneck), the same weights are reused (in the autoencoder) or a random initialization is used (the bottleneck classifier).

3.2 Finetuning

Network layouts for the methods compared in this work are illustrated in Fig. 1. Pretraining is done in exactly the same way for all architectures. The finetuning phases differ according to the cost function used and the number and structure of layers used in the decoder part of the network. Finetuning error is backpropagated through the network. In our experiments, conjugate gradient optimized was used for minimizing the error.

The bottleneck classifier produces an estimate $\tilde{y} \approx y$ when trained with input x and target y (y are class numbers, presented in 1-of- c binary encoding). The two-dimensional embedding for points x can be read from outputs of the bottleneck layer. After pretraining, two classifier layers (hidden and output) are added, with random initialization for the weights. Both layers have sigmoid activation functions. The outputs are combined using softmax, and the network is finetuned to minimize cross-entropy classification error.

The autoencoder produces a reconstruction $\tilde{x} \approx x$ when trained with input x and target x . Outputs of the middle layer form the low-dimensional embedding for points x . After pretraining, the decoding layers are added, mirroring the weights from pretrained layers. The network is finetuned to minimize mean square reconstruction error.

With NCA, kNN and t-SNE (which will be introduced in Sect. 4.1) cost functions, no new layers are added. Distances computed from the low-dimensional embedding e (see Fig. 1) are compared to distances between input vectors using the cost function, and the finetuning tries to minimize the cost.

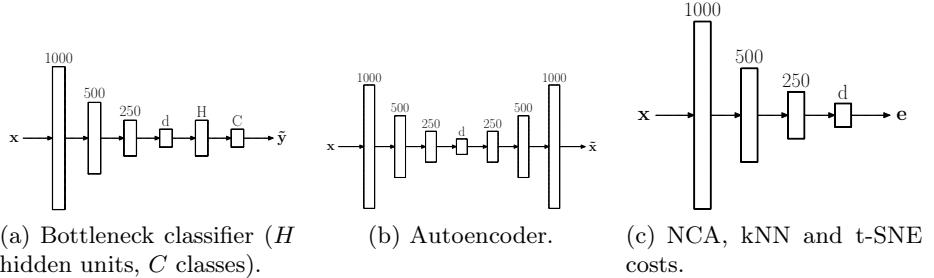


Fig. 1. Illustration of network layouts for the bottleneck classifier and comparison models, each performing dimension reduction to d dimensions. The text gives more details on training each network.

4 Comparison Methods

We compare the bottleneck classifier to two supervised cost functions (NCA and kNN), and to two unsupervised methods. To justify the often heavier computation required in the supervised setting, a supervised method should produce better results than a good unsupervised one. As t-SNE has achieved remarkably good results on several data sets, especially on clustered data, we include it in our comparison as the unsupervised baseline method. We also include autoencoders in the comparison because of their structural similarity to the bottleneck classifier.

In the following, we denote and index generic data points with h, i, j and l . N is the number of data points. The point i has coordinates \mathbf{x}_i in the data space and \mathbf{y}_i in the low dimensional space. Formulas below are taken from the cited references, but notation for different methods has been unified.

4.1 Stochastic Neighbor Embedding with Student-t Distributions

T-SNE [14] minimizes the Kullback-Leibler divergence $\sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$ between neighborhood probabilities p_{ij} in the data space and q_{ij} in the low dimensional space. Neighborhood probabilities in the data space are based on normal distributions,

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{h \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_h\|^2 / 2\sigma_i^2)}, \quad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}. \quad (1)$$

Variances σ_i vary according to data density, and are determined by perplexity parameter, which can be thought as a soft version of number of neighbors. In the low-dimensional space, neighborhood probabilities are computed using t-distribution, with a degrees-of-freedom parameter α ,

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{h \neq l} (1 + \|\mathbf{y}_h - \mathbf{y}_l\|^2/\alpha)^{-\frac{\alpha+1}{2}}} . \quad (2)$$

The heavy tails of the t-distribution compensate for the larger volume of a high-dimensional neighborhood. In the low-dimensional space, the points are allowed to move slightly further from each other than in the data space, but still the neighborhood relations are preserved well. A deep neural network which uses t-SNE cost function has been presented in [3].

4.2 Large Margin KNN Classification

A deep network for finding classification-preserving low-dimensional presentations is given in [5]. It applies, in a bit modified form, a cost function presented in [15]. Minimizing the cost function of [5]

$$\sum_{i=1}^N \sum_{j=1}^N \sum_{l=1}^N \eta_{il} \gamma_{ij} \cdot \max(0, 1 + \|\mathbf{y}_i - \mathbf{y}_l\|^2 - \|\mathbf{y}_i - \mathbf{y}_j\|^2) \quad (3)$$

maximizes the margin between classes by requiring that distance from point i to an other-class points j must be at least one plus the largest distance to k nearest same-class points j . The class relationships are encoded with the help of binary variables η_{il} (which is 1 for same-class point pairs i and l , if l is one of the k same-class neighbors chosen for comparison) and γ_{ij} (equals 1 if i and j is a chosen point from a class different than i).

Applying this idea directly would make the computational complexity $O(kN^2)$. Therefore, the computation is simplified by considering only m points from any other class. With C classes, this reduces the complexity to $O((C-1)kmN)$, but, as neighbors from all classes are needed for each point, the method is still quite slow.

4.3 Neighborhood Components Analysis

Neighborhood components analysis [8] learns a discriminative Mahalanobis metric by maximizing accuracy of nearest neighbor classification. The cost function measures expected number of correctly classified points, using soft neighbor assignments. Soft neighborhoods are defined as probabilities with which a point i chooses another point j as its nearest neighbor,

$$p_{ij} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)}{\sum_{h \neq i} \exp(-\|\mathbf{y}_i - \mathbf{y}_h\|^2)} , \quad p_{ii} = 0 . \quad (4)$$

In nearest neighbor classification this neighbor j determines the class for point i , and therefore the neighborhood probabilities depend on distances in the low-dimensional space. The closer the point i is located to other points of its class, the more probable it is to have its nearest neighbor from the correct class, and

the higher the value for the sum $\sum_{i=1}^N \sum_{j: c^i = c^j} p_{ij}$ (where c^i denotes the class of point i). The nonlinear version of NCA [4] finetunes a deep network to maximize this sum. Computational complexity is $O(N^2)$.

5 Experiments

5.1 Data Sets

We tried the bottleneck classifier, as well as the comparison methods, on three data sets. Dimension reduction was done to 30D, which would be appropriate when looking for low-dimensional but nevertheless classification-preserving representations of the data, and to 2D, which is easy to visualize.

The data sets used were the MNIST benchmark data on handwritten digits, USPS digits data, and a 100-dimensional variant of the 20 newsgroups data set.¹² The MNIST data contains 70000 digit images of 28x28 pixels, with roughly equal numbers of digits from each class. The data set has a fixed division into training data (60000 samples) and test data (10000 samples), which was respected also in our experiments. USPS data has 1100 16x16 pixel images from each class. These were used to form training data of 9000 samples and test data of 2000 samples. The newsgroups data consists of 16242 documents collected from internet newsgroups, with binary occurrence vectors for 100 chosen words. The documents are classified into four broad categories (comp.*, rec.*, sci.*, talk.*). We used 12000 randomly chosen documents for training and 4200 for testing (for easier division into training batches some documents were left out).

The training samples were further divided into training set proper (from now on, "training data" refers to this set) and validation data (10% of samples). The validation data was used for choosing parameter values and for determining the number of epochs in the finetuning phase.

5.2 Network Parameters

For RBM training and autoencoder finetuning we used the implementation of authors of [1].³ We implemented other cost functions based on the same code.

For the MNIST data, we used network layout of $784 \times 1000 \times 500 \times 250 \times d$, with d set to 2 or 30 as appropriate. In the 2D case we also experimented with $784 \times 500 \times 500 \times 2000 \times d$ layout often used in the literature [14], but due to larger number of weights it was slower to train and did not seem to improve the results, so a smaller net was chosen for the experiments. For the newsgroups data, the network layout was $100 \times 100 \times 75 \times 50 \times d$.

Literature value 50 epochs [1] was used in pretraining. Number of finetuning epochs was determined by early stopping, with upper limit of 100 epochs. Same pretraining result was used with different finetuning costs.

¹ All data sets are available from <http://www.cs.nyu.edu/~roweis/data.html>

² Available from <http://web.mit.edu/~rsalakhu/www/DBM.html>. We thank the authors for generously making their code available, which greatly facilitated our work.

5.3 Cost Function Parameters

Parameters for kNN cost were set to $k = 3$ and $m = 10$. Small values were chosen to keep the running times reasonable. Autoencoder and NCA cost have no tuning parameters. Degrees of freedom for t-SNE were $\alpha = d - 1$, one of the alternatives suggested in [3].

A preliminary run was done to check parameter sensitivities of t-SNE and the bottleneck classifier. The network was trained with the training data and results computed for the validation set. Ten uniformly spaced parameter values were tried and the one with best results was chosen for final experiments. Initial experimentation had shown little variation over repeated runs, so we decided not to do full cross-validation, which would have been computationally costly. Perplexity values $10 + 20j, j = 0 \dots 9$ were tried. In the final runs, we used perplexity values of 10, 50, 110, 10, 10, 130 MNIST/2D, MNIST/30D, news/2D, news/30D, USPS/2D, USPS/30D cases respectively.

For the bottleneck classifier, number of hidden units needs to be chosen. Values $10 + 40a, a = 0 \dots 9$ were tried. The chosen values were 210, 130, 10, 170, 10, 90 for MNIST/2D, MNIST/30D, news/2D, news/30D, USPS/2D, USPS/30D cases. In the 30D case, number of hidden units had no dramatic effect on accuracy: the differences between best and worst 1NN classification error were 0.6% (MNIST), 1.67% (newsgroups) and 1.33% (USPS). The 2D case had more variation, with differences 2.22%, 4.92% and 15.44%.

5.4 Semisupervised Experiment

Unlabeled data is often easier and cheaper to obtain in large quantities than labeled data. A benefit of using a generative model to initialize the network is that, even if the finetuning uses labels, the initialization phase can benefit also from the unlabeled samples. We performed a small experiment with semi-supervised learning, initializing the network with full training data but using varying fractions of data with labels in finetuning. The experiment was made on MNIST data set, with same parameters as with fully labeled data.

6 Results

The top row of Fig. 2 illustrates the baseline results: what we would achieve without supervision by class labels. Pretraining clearly gives a feasible starting point, with some structure visible but classes not clearly separable. Autoencoder does somewhat better, and t-SNE gives a good result, though with some overlap between classes. Visualizations using supervised methods are shown in the bottom row of Fig. 2. All supervised methods compared manage to separate the classes more clearly than the unsupervised methods.

As all the supervised methods aim at keeping the classes separate, classification accuracy is a natural criterion for numerical comparisons. We used k-NN classifier (for $k=1,3$) and a one-hidden-layer neural network classifier trained on

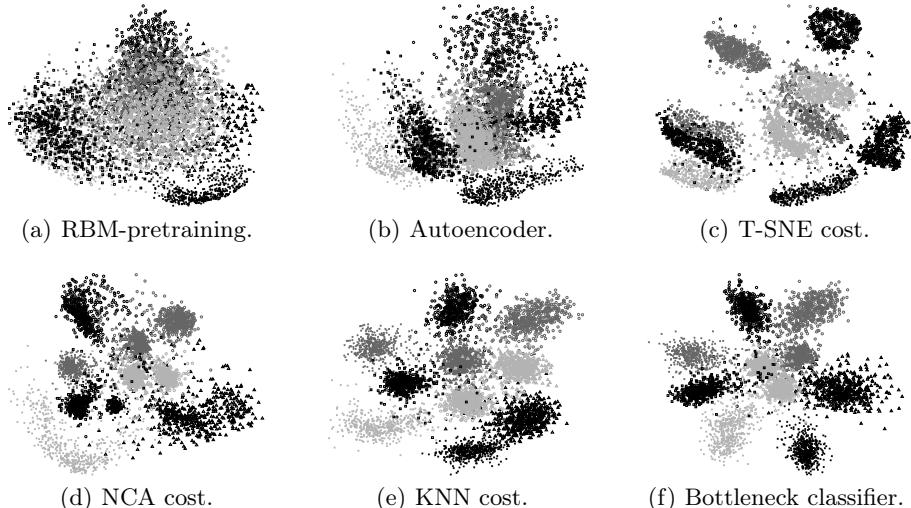


Fig. 2. Visualizations of the MNIST test data. The top row shows unsupervised and the bottom row supervised results. For readability, only half of the points have been plotted. The visualization does not show clear quality differences between the different supervised methods. For numerical comparisons, see tables in Sect. 6

the 2D/30D results produced by each network. Two methods were used, since k-NN classification might favor the neighborhood-based methods, whereas the bottleneck classifier could get some unfair benefit in network-based classification. Results are shown in Table 1 for the neural network classifier and in Table 2 for the k-NN classifier.

The bottleneck classifier performs consistently well in the 2D case, usually being the best method. In the 30D case, it typically falls second to one of the comparison methods while being superior to the other one. These results lead us to conclusion that performance of different supervised methods is somewhat data-dependent. It may also depend on parameters used, especially when increasing a parameter value would increase computational effort (as with the kNN cost, where relatively small numbers of neighbors were used).

Table 1. Classification errors (%) for the test data with a neural network classifier, which was trained on 2D/30D embeddings of the training data

		pretrain	autoenc	t-SNE	NCA	KNN	bottleclass
2D	MNIST	54.3	34.3	12.1	6.1	5.5	4.9
	news	43.4	30.8	30.0	22.0	25.5	21.9
	USPS	52.2	56.2	44.4	26.7	36.1	19.3
30D	MNIST	5.1	4.2	4.9	3.9	2.0	3.1
	news	22.4	21.8	23.0	18.7	22.1	20.0
	USPS	8.1	7.5	10.8	8.2	6.8	7.3

Table 2. Classification errors (%) for the test data with a k-NN classifier

		pretrain	autoenc	t-SNE	NCA	KNN	bottleclass
2D	MNIST 1-NN	61.8	40.5	14.5	8.3	8.1	6.91
	3-NN	60.9	37.9	10.6	6.6	6.3	5.43
	news	41.0	33.5	32.5	25.93	30.7	27.7
		40.0	30.9	30.7	23.05	27.1	24.2
	USPS	60.2	64.8	40.4	28.3	48.6	27.80
		59.1	60.6	36.5	25.0	42.5	21.75
30D	MNIST 1-NN	3.6	3.2	3.9	1.9	1.88	1.9
	3-NN	3.2	2.8	3.6	1.8	1.47	1.8
	news	28.9	28.5	27.2	22.62	25.1	24.9
		27.6	26.8	25.3	20.29	23.0	22.3
	USPS	9.8	8.4	10.4	5.40	5.8	5.9
		9.8	8.6	10.1	5.7	5.35	6.5

Table 3. K-NN classification errors (%) for the test data in the semisupervised experiment (MNIST data), for different amounts of label information

	1 %	5 %	10 %	30 %	50 %	70 %	90 %
2D	1-NN	42.4	26.3	19.8	11.4	10.2	8.1
	3-NN	39.1	22.1	16.7	9.2	7.7	6.3
30D	1-NN	3.3	3.0	2.7	2.3	2.0	1.8
	3-NN	3.0	2.6	2.5	2.1	1.8	1.6

Based on these experiments, no clear order of superiority could be established between the three methods. Therefore, issues like computational complexity become important when choosing the method. The bottleneck classifier is worth considering because of its computational simplicity. Also, if main interest lies at the 2D visualizations, the bottleneck classifier could be the best choice.

Results for the semi-supervised experiment are shown in Table 3. In the 2D case, accuracy gained by adding labels seems to saturate rapidly: difference between 30% and 90% of labels is only about 4 percentage units, much less than the about 8 percentage unit difference between 10% and 30% of labels. In the 30D case, adding more labels improves results roughly linearly.

7 Conclusions

We presented a method for introducing supervision into an RBM-pretrained deep neural network which performs dimension reduction by forcing information to flow through a bottleneck layer. The idea presented is very simple: after pretraining, the network is trained as a classifier. The method is simple to implement and scales linearly with the number of data points.

We compared our method with two other supervised methods. We found that the bottleneck classifier gives results comparable to the other methods. We

therefore see it as a choice worth recommending, since it is computationally much lighter than the other methods.

In this work we trained the network as a multiclass classifier. It could equally easily be trained to perform regression. It is a possible direction for future work to test the applicability of the idea for sufficient dimension reduction (see e.g. [16] and references therein), which aims to find low-dimensional presentations preserving the information necessary for predicting a target variable in regression.

References

1. Hinton, G.E., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. *Science* 313, 504–507 (2006)
2. Mao, J., Jain, A.K.: Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks* 6, 296–317 (1995)
3. van der Maaten, L.: Learning a parametric embedding by preserving local structure. In: Proc. of AISTATS. JMLR: W&CP 5, vol. 5, pp. 384–391 (2009)
4. Salakhutdinov, R., Hinton, G.: Learning a nonlinear embedding by preserving class neighborhood structure. In: Proc. of AISTATS (2007)
5. Min, R., Stanley, D.A., Yuan, Z., Bonner, A., Zhang, Z.: A deep non-linear feature mapping for large-margin kNN classification. In: Proc. of ICDM (2009)
6. Elman, J.L., Zipser, D.: Learning the hidden structure of speech. *Journal of the Acoustic Society of America* 83, 1615–1626 (1988)
7. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. *Journal of Machine Learning Research* 10, 1–40 (2009)
8. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Proc. of NIPS, vol. 17, pp. 513–520 (2005)
9. Kurtz, K.J.: The divergent autoencoder (DIVA) model of category learning. *Psychonomic Bulletin & Review* 14, 560–576 (2007)
10. Weston, J., Ratle, F., Collobert, R.: Deep learning via semi-supervised embedding. In: Proc. of ICML (2008)
11. Ranzato, M., Szummer, M.: Semi-supervised learning of compact document representations with deep networks. In: Proc. of ICML (2008)
12. Intrator, N., Edelman, S.: Learning low-dimensional representations via the usage of multiple-class labels. *Network: Computation in Neural Systems* 8, 259–281 (1997)
13. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. In: Rumelhart, D.E., McClelland, J.L. (eds.) *Parallel Distributed Processing*, vol. 1, pp. 194–281. MIT Press, Cambridge (1986)
14. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 2579–2605 (2008)
15. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10, 207–244 (2009)
16. Kim, M., Pavlovic, V.: Covariance operator based dimensionality reduction with extension to semi-supervised learning. In: Proc. of AISTATS. JMLR: W&CP 5, vol. 5, pp. 280–287 (2009)

Local Modeling Classifier for Microarray Gene-Expression Data[☆]

Iago Porto-Díaz, Verónica Bolón-Canedo, Amparo Alonso-Betanzos,
and Óscar Fontenla-Romero

Department of Computer Science, University of A Coruña, Spain
`{iporto,vbolon,ciamparo,ofontenla}@udc.es`

Abstract. Gene-expression microarray is a novel technology that allows to examine tens of thousands of genes at a time. For this reason, manual observation is not feasible anymore and machine learning methods are progressing to analyze these new data. Specifically, since the number of genes is very high, feature selection methods have proven valuable to deal with this unbalanced – high dimensionality and low cardinality – datasets. Our method is composed by a discretizer, a filter and the FVQIT (Frontier Vector Quantization using Information Theory) classifier. It is employed to classify eight DNA gene-expression microarray datasets of different kinds of cancer. A comparative study with other classifiers such as Support Vector Machine (SVM), C4.5, naïve Bayes and k-Nearest Neighbor is performed. Our approach shows excellent results outperforming all other classifiers.

1 Introduction

In genetics, gene expression is the process by which genes are used to synthesize proteins and other gene products such as ribosomal RNA or transfer RNA. Being able to quantify the levels at which genes are expressed within cells is very important and allows applications such as identifying viral infections of cells, determining susceptibility to cancer or finding if a bacterium is resistant to penicillin.

The traditional approach in molecular biology has been to locally examine a single gene at a time. Several techniques can be used, but they quickly become inoperable as the number of genes in the study increases. Moreover, in recent years the volume of data available in genomics has increased in several orders of magnitude. Information coming from large-scale genome sequence projects is attracting the interest to the study of global mechanisms.

DNA microarrays are a novel technology used to measure gene expression [1]. The idea behind microarrays is to measure the amount of mRNA present

[☆] This work was supported in part by Xunta de Galicia under Project Code 08TIC012105PR and under the program “Axudas para a consolidación e a estruturación de unidades de investigación competitivas” (code 2007/134), and by Spanish Ministerio de Ciencia e Innovación under Project Code TIN2009-10748. These last two are partially supported by the European Union ERDF.

in a cell, thus estimating the expression levels of the genes responsible for the synthesis of these molecules. Structurally, DNA microarrays consist of a series of single stranded DNA oligonucleotides called probes, each of them containing picomoles – 10^{-12} moles – of DNA of a complementary sequence to the mRNA molecule of the gene that it is targeting. Fluorescently tagged mRNA molecules should hybridise to the probes they are complementary of, in such a way that the level of fluorescence of the dye reveals the amount of hybridisation that has taken place. When examining this with a scanner, a text file containing relevant data such as the level of fluorescence of each spot and the level of background noise can be output and analyzed with a computer afterwards. Further details of the technologies can be seen in [2].

At the same time as the development of the DNA microarrays technologies, machine learning methods have progressed to analyze these new data [3,4,5,6]. DNA microarrays pose some major issues to be addressed by the researchers, such as:

- The technical process may add noise to data.
- Genetic variability may cause the gene expressions of two characters to be different.
- Microarray datasets have high dimensionality (thousands or tens of thousands of features) and very few samples (less than a hundred), which makes difficult a good generalization.

Having more features than instances causes trouble to machine learning methods because they can not generalize properly. Furthermore, most genes in a DNA microarray are not relevant for most common problems such as molecular classification of cancer [7]. As of this, feature selection helps machine learning methods to deal with those particular unbalanced problems and helps biologists to understand relations between gene expression and diseases. To gloss over these defects, feature selection has been tried in several works. In [8] several filters and wrappers combined with several classifiers have been used. In [9] a new wrapper named BIRS has proven to be more efficient than other wrapper methods. In [10] 10-fold cross-validation is recommended over leave-one-out when dealing with microarray data because the selection bias is assessed and corrected. The research carried out in [11] highlights the importance of including a discretization step in order to obtain better classification performances. In [12] a review of feature selection techniques for microarray analysis is shown.

In this paper, we are proposing a method composed by a discretizer, a filter, and the FVQIT [13] classifier. This method is employed to classify DNA gene-expression microarray datasets of different kinds of cancer. A comparative study with other methods such as Support Vector Machine (SVM), C4.5, naïve Bayes and k-Nearest Neighbor is performed. In Sect. 2 the proposed method is fully described. Section 3 describes the experiment carried out and the results obtained. Finally, Sect. 4 contains the conclusions obtained and the future work.

2 Method

There exist two main kinds of feature selection methods: wrappers and filters. On one hand, wrapper methods use an inductive algorithm to estimate the values of the different subsets. On the other hand, filter methods select gene subsets by using an evaluation function that relies on properties of data and is independent of any algorithm. By this reason – independence from the classifier –, we have chosen filter methods over wrappers. Since some features of the microarray datasets show unbalanced values and several filters need discrete data to work, a previous stage of discretization is advised. Thus, in this work, two discretizers (EMD and PKID) and three filter methods (CFS, Consistency-based and INTERACT) will be utilized along with a classifier named FVQIT (Frontier Vector Quantization using Information Theory), as can be seen in Fig. II. Several discretizers and filters have been chosen in order to handle a wider range of comparison.

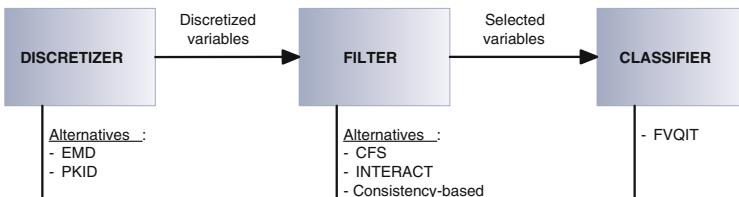


Fig. 1. Structure of the proposed method

2.1 Discretizers

- EMD (Entropy Minimization Discretization) [14] considers the midpoint between each successive pair of sorted values (x_j, x_{j+1}) as a candidate cut-point. For each candidate cut-point, all data left at both sides are discretized into two intervals and the resulting class information entropy is calculated. The cut-point for which the entropy is minimum is selected. The process is repeated recursively, hence selecting the minimum-entropy cut-points. The Minimum Description Length principle (MDL) is utilized as a stopping criterion.
- PKID (Proportional k-Interval Discretization) [15] tries to seek an appropriate trade-off between the bias and the variance of the probability estimation by adjusting the number and size of discretization intervals to the number of training instances. Thus, a numerical attribute X_i is discretized into \sqrt{n} intervals, each of them containing \sqrt{n} instances, where n is the number of training instances with known values $\{x_1, \dots, x_n\}$ for the attribute X_i .

2.2 Filter Methods

- CFS (Correlation-based Feature Selection) [16] ranks feature subsets according to a heuristic evaluation function based on correlation. The higher the

correlation with the class and the lower the correlation with other features the better the value of the evaluation function.

- Consistency-based Filter [17] judges the value of a subset of features according to the level of consistency of class values depending on training data. A random subset of features is generated in each iteration and checked against an inconsistency criterion. If the inconsistency rate is lower than a given threshold, the subset becomes the current minimum. The inconsistency criterion is the key for the success of this algorithm. It specifies a quality measure that determines the maximum reduction of the dimensionality of data.
- INTERACT [18] is based on symmetrical uncertainty (SU), which is defined as the ratio between the Information Gain (IG) and the Entropy (H) of two features, x and y :

$$SU(x, y) = 2 \frac{IG(x/y)}{H(x) + H(y)} \quad (1)$$

where $IG(x/y) = H(y) + H(x) - H(x, y)$ is the Information Gain, $H(x)$ is the Entropy, and $H(x, y)$ is the Joint Entropy.

Moreover, INTERACT uses the consistency contribution (C-contribution), which is an indicator about how the elimination of a feature would affect consistency. Thus, by using SU and C-contribution, INTERACT can handle feature interaction and select relevant features.

2.3 Machine Learning Classifier

The classification method used is a supervised architecture for binary classification. It is based on local modeling and information theory [13]. The algorithm works in two stages, as shown in Fig. 2. First, a set of nodes is set on the frontier between classes in such a way that each of these nodes defines a local model. Last, an one-layer neural network is trained in each local model.

- Stage 1: Selection of Local Models. Local models are constructed by applying the FVQIT algorithm. The objective is to place a set of nodes on the frontier between the two classes, in such a way that each node represents a local model. Both data points and nodes are considered two kinds of particles with a potential field associated. These fields induce repulsive and attractive interactions between particles, depending on its sign. Data particles belonging to different classes have different signs. In this context, the algorithm minimizes a function of energy that evaluates the divergence between the Parzen density estimator of data points $f(x)$ and the estimator of the distribution of the nodes $g(x)$, where $f(x)$ and $g(x)$ are:

$$\begin{aligned} f(x) &= \frac{1}{N} \sum_{i=1}^N K\left(x - x_i, \sigma_f^2\right) \\ g(x) &= \frac{1}{N} \sum_{i=1}^N K\left(x - w_i, \sigma_g^2\right) \end{aligned} \quad (2)$$

where N is the number of data points, K is any kernel function, σ_f^2 and σ_g^2 are the variances of the kernel functions, $x_i \in \mathbb{R}^n$ are data points, and $w_i \in \mathbb{R}^n$ are the weights associated to the nodes.

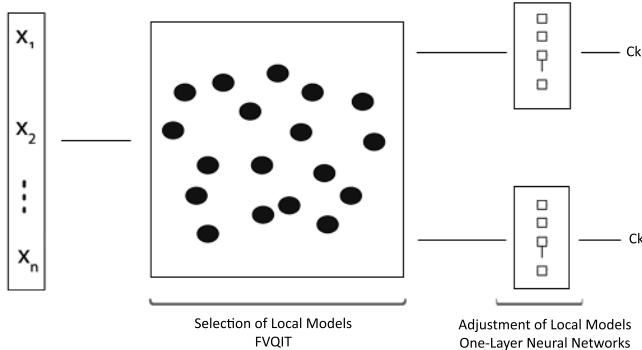


Fig. 2. Structure of the FVQIT Architecture

The function of energy $J(w)$ to be minimized is:

$$J(w) = 2 \log \int f^+(x) g(x) dx - 2 \log \int f^-(x) g(x) dx + \log \int g^2(x) dx \quad (3)$$

where $f^+(x)$ and $f^-(x)$ are the estimators of the distributions of data of both classes. In order to minimize $J(w)$ in (3), the algorithm tries to: minimize the cross-correlation between distributions of nodes and distributions of data of one class and maximize the cross-correlation between distributions of nodes and distributions of data of the other class. In this way, the nodes are placed along the frontier between classes. In case of overlapping classes, the nodes will tend to be situated halfway between them.

Using the gradient descent method, the weight update rule for a node k becomes:

$$w_k(n+1) = w_k(n) - \eta \left(\frac{\Delta V}{V} + \frac{\Delta C_+}{C_+} - \frac{\Delta C_-}{C_-} \right) \quad (4)$$

where $C_+ = \int f^+(x) g(x) dx$ is the contribution of data of one class to $J(w)$; $C_- = \int f^-(x) g(x) dx$ is the contribution of the data of the other class, and $V = \int g(x)^2 dx$ is the contribution of the interactions among nodes, n denotes the learning epoch, η the learning rate, and Δ indicates “derivative with respect to w_k ”.

- Stage 2: Adjustment of Local Models. Data is contained in the local models created in the previous stage, in such a way that each local model comprises the closest data using the Euclidean distance. Thus, the input space is completely filled by the set of local models, as input data are always assigned to a local model. Since nodes define the frontier between classes, each local model adjusts a classifier for the data points in its region of influence.

The local classifier is a one-layer neural network, because local modeling algorithms may suffer from temporal efficiency problems, caused by the process of training several local classifiers. Thus, a lightweight classifier as the one-layer neural network trained with the efficient algorithm presented

in [19] has proved suitable. This algorithm obtains the weights of a neuron with a complexity of $O((I+1)^2)$, where I is the dimension of the input vectors.

After the training process, when a new pattern needs to be classified, the method first calculates the closest node w_k to x_n using the Euclidean distance and then classifies x_n using the neural network associated to the local model w_k .

3 Experiment

The proposed method has been tested over eight binary DNA microarray datasets. All datasets utilized in this work are publicly available at [20]. The method chosen to estimate the real error of the model is based on a partition in three sets. Some datasets are originally divided in training and test, but some are not. So first, in order to compare our results with previous authors, CNS, DLBCL, Colon, Ovarian and Breast datasets have been divided as in [11], using 2/3 for training and 1/3 for testing. Upon the training sets, a 10-fold cross-validation is performed, in order to estimate the validation error to choose a good configuration of parameters for the algorithm. 10-fold cross-validation has been chosen over leave-one-out cross-validation because the variance of the latter can be too high [10]. The number of features and the division on training and test sets for each dataset is shown in Table [11].

The results of the proposed method have been compared with those obtained by other classifiers, such as decision trees (C4.5), naïve Bayes (NB), k-Nearest Neighbor (k-NN) and Support Vector Machines (SVM). The results of the first three classifiers have been obtained from [11]. Table [2] shows the estimated test errors – in percentage – and the number of features selected by each method tested. Despite having executed all six combinations of discretizer + filter, only the best result for each classifier in each dataset is shown. Also, the best result obtained for each dataset is emphasized in bold font.

As can be seen in Table [2], the FVQIT method obtains good performance in all datasets, with an adequate number of selected features. In Table [3] a ranking

Table 1. Description of the datasets

Dataset	No. features	Total samples	Training set	Test set
Breast [21]	24 481	97	78	19
CNS [22]	7129	60	40	20
Colon [23]	2000	62	42	20
DLBCL [24]	4026	47	32	15
Leukemia [7]	7129	72	38	34
Lung [25]	12 533	181	32	149
Ovarian [26]	15 154	253	169	84
Prostate [27]	12 600	136	102	34

Table 2. Best estimated test errors and number of features selected

Dataset	Method	Test Error (%)	No. of features
Breast	PKID+CFS+FVQIT	21.05	17
	EMD+CFS+SVM	21.05	119
	EMD+Cons+NB	26.32	5
	EMD+Cons+1-NN	26.32	5
	PKID+INT+C4.5	21.05	3
CNS	PKID+INT+FVQIT	25	4
	EMD+CFS+SVM	30	60
	PKID+INT+NB	25	4
	PKID+INT+1-NN	35	4
	EMD+INT+C4.5	35	47
Colon	PKID+CFS+FVQIT	10	12
	PKID+CFS+SVM	10	12
	EMD+Cons+NB	15	3
	EMD+Cons+1-NN	15	3
	EMD+Cons+C4.5	15	3
DLBCL	EMD+INT+FVQIT	0	36
	EMD+INT+SVM	6.67	36
	EMD+INT+NB	6.67	36
	EMD+INT+1-NN	33.33	36
	EMD+Cons+C4.5	13.33	2
Leukemia	PKID+Cons+FVQIT	0	2
	PKID+CFS+SVM	2.94	18
	PKID+CFS+NB	5.88	18
	EMD+Cons+1-NN	8.82	1
	PKID+Cons+C4.5	5.88	2
Lung	EMD+INT+FVQIT	0.67	40
	PKID+INT+SVM	1.34	40
	EMD+Cons+NB	4.70	1
	PKID+INT+1-NN	0	40
	EMD+Cons+C4.5	18.12	1
Ovarian	PKID+INT+FVQIT	0	3
	EMD+Cons+SVM	0	3
	EMD+Cons+NB	0	3
	EMD+Cons+1-NN	0	17
	EMD+Cons+C4.5	1.19	3
Prostate	PKID+CFS+FVQIT	5.88	11
	PKID+INT+SVM	73.53	3
	PKID+Cons+NB	26.47	2
	PKID+Cons+1-NN	26.47	2
	PKID+Cons+C4.5	26.47	2

of the performance results of all the methods compared is shown. The ranking assigns a position between 1 and 5 to each method in each dataset, taking into account the ties between them.

Table 3. Position of each method in the comparative study

Classifier	Breast	CNS	Colon	DLBCL	Leukemia	Lung	Ovarian	Prostate	Average
FVQIT	1st	1st	1st	1st	1st	2nd	1st	1st	1.125
SVM	1st	3rd	1st	2nd	2nd	3rd	1st	5th	2.25
NB	4th	1st	3nd	2nd	3rd	4th	1st	2nd	2.5
1-NN	4th	4th	3nd	5th	5th	1st	1st	2nd	3.125
C4.5	1st	4th	3nd	4th	3rd	5th	5th	2nd	3.375

In Table 3 the proposed method obtains the best results for all datasets but the Lung Cancer dataset. In this case, the difference with the best method is lower than 1%. In average, the proposed method is clearly preferable above the other methods studied.

In light of the above, we can conclude that the classifier FVQIT is the best classifier to be combined with discretizers and filters to deal with problems with a much higher number of features than instances, such as DNA microarray gene-expression problems.

4 Conclusions and Future Work

In this paper, eight DNA microarray gene-expression datasets have been classified using a method that combines a discretizer, a filter and a local classifier called FVQIT. The classifier used is capable of obtaining complex classification models by constructing a piecewise borderline between the different regions and then classifying locally by using one-layer neural networks. Two discretizers and three filter methods have been checked, and the proposed classifier has been compared with other classifiers such as Support Vector Machines (SVM), naïve Bayes (NB), C4.5, and k-Nearest Neighbor (k-NN).

The proposed method is able to successfully classify datasets with high dimensionality in number of features and few instances. Some methods can not even confront those datasets due to the very high number of features (tens of thousands) and other suffer from overfitting problems when the ratio number of samples / number of features is small. The results obtained show that the proposed method obtains better performance than the other methods. In average, the proposed algorithm is the best option, obtaining an average of 1.125 position in the ranking of classifiers, clearly improving the 2.25 of the second classified.

As future work, we plan to expand the proposed method to be able to deal with multiclass problems and apply it to some multiclass microarray-based datasets.

References

1. Chee, M., Yang, R., Hubbell, E., Berno, A., Huang, X.C., Stern, D., Winkler, J., Lockhart, D.J., Morris, M.S., Fodor, S.: Accessing Genetic Information with High-Density DNA Arrays. *Science* 274(5287) (1996)

2. Eisen, M.B., Brown, P.O.: DNA Arrays for Analysis of Gene Expression. *Methods in Enzymology*, pp. 179–204. Academic Press Inc. Ltd., London (1999)
3. Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., Yakhini, Z.: Tissue Classification with Gene Expression Profiles. *Journal of Computational Biology* 7(3-4), 559–583 (2000)
4. Brown, M.P.S., Grundy, W.N., Lin, D., Cristianini, N., Sugnet, C.W., Furey, T.S., Ares, M., Haussler, D.: Knowledge-Based Analysis of Microarray Gene Expression Data by Using Support Vector Machines. *Proceedings of the National Academy of Sciences* 97(1) (2000)
5. Der, S.D., Zhou, A., Williams, B.R.G., Silverman, R.H.: Identification of Genes Differentially Regulated by Interferon α , β , or γ Using Oligonucleotide Arrays. *Proceedings of the National Academy of Sciences* 95(26) (1998)
6. Lim, S.M., Johnson, K.F.: Methods of Microarray Data Analysis. In: *Proceedings of the First Conference on Critical Assessment of Microarray Data Analysis CAMDA 2000*. Kluwer Academic Publishers, Dordrecht (2001)
7. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science* 286(5439) (1999)
8. Wang, Y., Tetko, I.V., Hall, M.A., Frank, E., Facius, A., Mayer, K.F.X., Mewes, H.W.: Gene Selection from Microarray Data for Cancer Classification. A Machine Learning Approach. *Journal of Computational Biology and Chemistry* 29(1), 37–46 (2005)
9. Ruiz, R., Riquelme, J.C., Aguilar-Ruiz, J.S.: Incremental Wrapper-Based Gene Selection from Microarray Data for Cancer Classification. *Pattern Recognition* 39(12), 2383–2392 (2006)
10. Ambroise, C., McLachlan, G.J.: Selection Bias in Gene Extraction on the Basis of Microarray Gene-Expression Data. *Proceedings of the National Academy of Sciences* 99(10), 6562–6566 (2002)
11. Bolón-Canedo, V., Sánchez-Marcano, N., Alonso-Betanzos, A.: On the Effectiveness of Discretization on Gene Selection of Microarray Data. In: *Proceedings of International Joint Conference on Neural Networks, IJCNN* (in press, 2010)
12. Saeys, Y., Inza, I., Larranaga, P.: A Review of Feature Selection Techniques in Bioinformatics. *Bioinformatics* 23(19), 2507–2517 (2007)
13. Martinez-Rego, D., Fontenla-Romero, O., Porto-Díaz, I., Alonso-Betanzos, A.: A New Supervised Local Modelling Classifier Based on Information Theory. In: *Proceedings of the International Joint Conference on Neural Networks, IJCNN*, pp. 2014–2020 (2009)
14. Fayyad, U.M., Irani, K.B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In: *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1022–1029. Morgan Kaufmann, San Francisco (1993)
15. Yang, Y., Webb, G.I.: Proportional k-Interval Discretization for Naïve-Bayes Classifiers. In: Flach, P.A., De Raedt, L. (eds.) *ECML 2001. LNCS (LNAI)*, vol. 2167, pp. 564–575. Springer, Heidelberg (2001)
16. Hall, M.A.: Correlation-Based Feature Selection for Machine Learning. PhD Thesis, University of Waikato, Hamilton, New Zealand (1999)
17. Dash, M., Liu, H.: Consistency-Based Search in Feature Selection. *Artificial Intelligence Journal* 151(1-2), 155–176 (2003)
18. Zhao, Z. and Liu H.: Searching for Interacting Features. In: *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI*, pp. 155–176 (2003)

19. 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)
20. Ridge, K.: Kent Ridge Bio-Medical Dataset (2009),
<http://datam.i2r.a-star.edu.sg/datasets/krbd> (Last access: March 2010)
21. Van't Veer, L.J., Dai, H., Van de Vijver, M.J., et al.: Gene Expression Profiling Predicts Clinical Outcome of Breast Cancer. *Nature* 415(6871), 530–536 (2002)
22. Pomeroy, S.L., Tamayo, P., Gaasenbeek, P., et al.: Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression. *Nature* 415(6870), 436–442 (2002)
23. Alon, U., Barkai, N., Notterman, D.A., Gish, K., et al.: Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. *Proceedings of the National Academy of Sciences* 96(12), 6745–6750 (1999)
24. Alizadeh, A.A., Elisen, M.B., Davis, R.E., et al.: Distinct Types of Diffuse Large B-Cell Lymphoma Identified by Gene Expression Profiling. *Nature* 403(6769), 503–511 (2000)
25. Gordon, G.J., Jenson, R.V., Hsiao, L.L., et al.: Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer and Mesothelioma. *Cancer Research* 62(17), 4963–4967 (2002)
26. Petricoin, E.F., Ardekani, A.M., Hitt, B.A., et al.: Use of Proteomic Patterns in Serum to Identify Ovarian Cancer. *The Lancet* 359(9306), 572–577 (2002)
27. Singh, D., Febbo, P.G., Ross, K., Jackson, D.G., et al.: Gene Expression Correlates of Clinical Prostate Cancer Behavior. *Cancer Cell* 1(2), 203–209 (2002)

Learning of Lateral Connections for Representational Invariant Recognition

Christian Keck and Jörg Lücke

Frankfurt Institute for Advanced Studies, Goethe-Universität Frankfurt,
Ruth-Moufang-Str. 1, 60438 Frankfurt am Main, Germany
{keck,luecke}@fias.uni-frankfurt.de

Abstract. We study an artificial neural network that learns the invariance properties of objects from data. We start with a bag-of-features encoding of a specific object and repeatedly show the object in different transformations. The network then learns unsupervised from the data what the possible transformations are and what feature arrangements are typical for the object shown. The information about transformations and feature arrangements is hereby represented by a lateral network of excitatory connections among units that control the information exchange between an input and a down-stream neural layer. We build up on earlier work in this direction that kept a close relation to novel anatomical and physiological data on the cortical architecture and on its information processing and learning. At the same time we show, based on a new synaptic plasticity rules, that learning results in a strong increase of object finding rates in both artificial and more realistic experiments.

1 Introduction

Object recognition based on visual information is a challenging task for both biological networks and artificial systems. Artificial neural networks have been applied to the task of visual object recognition for many decades and have used different strategies. Although much can be achieved by pure feed-forward processing (e.g., perceptron-like approaches [12]), systems that recurrently integrate information (laterally and across hierarchical stages) have recently come into focus (e.g., [34]). Neurophysiological and neuroanatomical data strongly support the importance of lateral and recurrent processing. Technical systems have very much evolved to architectures and methodologies that show features similar to those of biological systems. Many algorithms use Gabor wavelets transformations or similar preprocessing to represent visual features, they learn from data, and they recurrently integrate information.

One possible role of recurrent information integration is hereby the routing of bottom-up information [56] for further processing. In recent work on such systems [78] it was shown that lateral information exchange can integrate information about feature similarities and feature arrangements to find the corresponding points between an input image and an image in memory. Based on a network of balanced excitation and inhibition [9] such a routing was shown to

be very fast with routing being accomplished in an equivalent of less than 100ms in biological time. Rapid routing in these systems was enabled through control units (compare, e.g., [5][10]) that are kept in balanced states [7]. Integration of feature arrangements and feature similarities was in [7] enabled by a lateral network of excitatory connections between control units. In earlier work, these connections were fixed to fit the two-dimensional topology of natural images. In this paper we show that such connections can be learned from data. We will keep the basic architecture and building blocks of earlier work [7] to maintain the close relation of the system to neuroanatomical entities and neurophysiological results. However, we will study a new learning dynamics that shows robust results on artificial and more realistic data.

2 A Model of Cortical Columns

In [9][11][12] a model of the cortical column was described that mirrors recent anatomical results on cortical microcircuits (e.g. [13]). The model consists of inhibitorily coupled sub-networks (or *sub-units*) of excitatory neurons. Neurons within the sub-units are excitatorily connected to afferent input and to input from other neurons of the sub-unit but not to neurons of other sub-units (see [9] for more details). Inhibition between sub-units leads to competition among all sub-units within a column. The specific type of competition is hereby crucial for the processing and learning capabilities of the column. Winner-take-all (WTA) competition or soft versions (e.g., soft-max [14]) are, for instance, closely associated with systems for clustering. To learn the components of data, e.g. the components of visual patches, a soft-k-WTA competition has proven beneficial (see [15] for a relation to functional approaches).

In the column model [9], a soft-k-WTA competition is realized through an inhibition cycle that, for each input, increases competition between the sub-units in time. Learning will later take place during this increase which realizes the soft-k-WTA competition. The dynamics of a single sub-unit p_α is described by:

$$\frac{d}{dt} p_\alpha = f(p_\alpha, \nu \max_{\beta=1,\dots,k} \{p_\beta\}) + \kappa \mathcal{J}_\alpha + \sigma \eta_t, \quad (1)$$

$$f(p_\alpha, h) = a p_\alpha (p_\alpha - h - p_\alpha^2). \quad (2)$$

The change of the activity p_α is affected by the inhibition ν times the maximum activity of all sub-units of the column and the external input \mathcal{J}_α , which is weighted with a coupling factor κ . The noise-term of the stochastic differential equation is represented by $\sigma \eta_t$. In the beginning of a computation done by the sub-units of the column all sub-units are active and the inhibition-factor ν is at a low level. We then increase ν in a linear fashion to let the nonlinear competition of the sub-units take more and more effect. In this way the column as a collection of competing sub-units performs a soft-k-WTA computation. The term $\max_{\beta=1,\dots,k} \{p_\beta\}$ let's each subunit directly compete with the strongest sub-unit of the column. In combination with the inhibition this is the driving force for the soft-k-WTA computation. The properties of a single column were studied

more extensively in [9]. Here we use such columns as basic computational units to form a network (compare [7]).

3 Correspondence Finding in a Network of Cortical Columns

Image preprocessing with 2D Gabor filters. We use a 2D-Gabor-filter to extract local frequency-information as preprocessing step. The use of Gabor filters is biological motivated: cells with Gabor-like responses were found in V1 of mammals (e.g. [16]). From the functional point of view preprocessing with Gabor filters is also reasonable: it optimizes space-frequency resolution with respect to the uncertainty principle. The Gabor transformation is a windowed fourier transformation (WFT) with a gaussian window. For details see [4]. We here use 5 scales and 8 orientations and call this collection of $5 \times 8 = 40$ features a feature-vector. To visualize these 40 dimensional vectors we use a circular plot shown in figure 11. High frequencies are shown in the inner circles, low frequencies in the outer circles.

The architecture of the network. Finding corresponding points between two images is a key ability in order to apply model-based object recognition. While systems working with a "bag of features" do not integrate information about feature-positions, so called correspondence based systems (e.g. [17]) have an explicit model of the feature-layout describing an object. We here use an image showing the object (called the model image) and a larger input image showing the object at a random position in front of a background. Instead of finding corresponding pixels we want to find corresponding feature-vectors given by the Gabor filter responses. We therefore define a graph (here: grid-graph) of points for feature extraction for the model. In the same way but with proportionally more points we do the same for the input image. After feature preprocessing with Gabor filters we obtain a 40 dimensional (8 orientations times 5 scales) vector for each graph-node. The amount of overall possible correspondences is the multiplication of the number of feature-vectors of model and input. Given the graph of feature vectors for input and model we want the network to evaluate all possible correspondences. Let us focus on a single model feature vector: all input feature vectors represent possible corresponding points, so there is competition among the input feature vectors being the corresponding one. From this point of view we design our network of columns in the following intuitive way: each column represents a node on the model graph. Each sub-unit of a column then represents a node on the input graph. This resulting network has a clearly defined number of columns all having the same amount of sub-units. In this way we use the columns and their sub-units to control the finding of the correct correspondence in the context of all possible correspondences. We therefore name the columns *control columns* and the sub-units control units (compare, e.g., [5][10]). Figure 11 shows a simplified version with one-dimensional images for model and input.

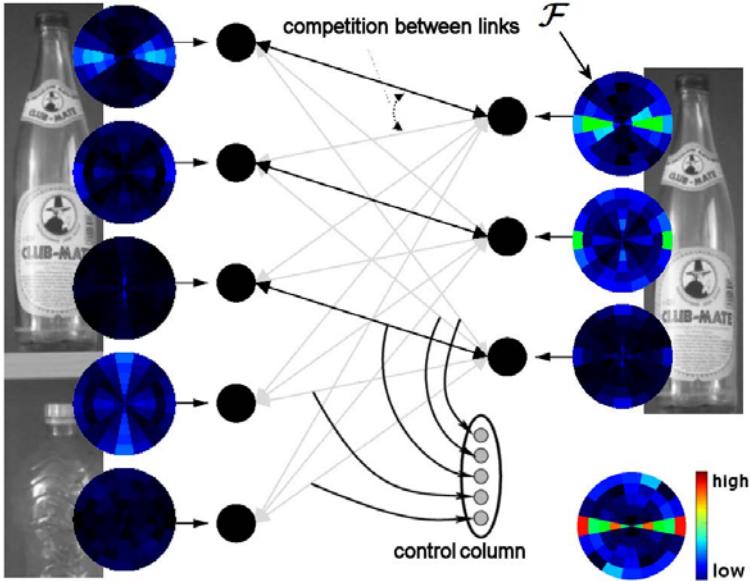


Fig. 1. Concept of the correspondence finding network: We apply columns to find the matching correspondences. Each possible link to a feature-vector \mathcal{F} on the input-image is represented by a control unit of the control column. All possible correspondences are visible here, the correct ones are drawn in black. For the feature vector at the bottom of the model a sketch of the control column with its 5 control units is drawn. The arrows depict the flow of input to the control units.

Input to control units. We just defined a network of control columns consisting of control units in order to find the corresponding feature-vectors. So far we did not define the input that control units receive. As shown in equation 11 the control units get external input \mathcal{J} . Equation 3 shows how it is composed of direct input from similarity between input- and model-feature-vectors (5) and input from lateral connections representing topological information (4). $\mathcal{L}_{\alpha\beta}^{cc'}$ is the strength of the lateral connection between the units. The parameter C_I is used for balancing the two different input-sources. Note that in contrast to [7][8] we here do not use columns for similarity computation. We have chosen to use a more direct approach to evaluate feature similarity (compare e.g. [6]) while, e.g., [8][7] uses a neural similarity evaluation.

$$\mathcal{J}_\alpha^c = C_I S(\mathcal{F}_c^M, \mathcal{F}_\alpha^I) + (1 - C_I) \mathcal{J}_{\mathcal{L}_\alpha^c} \quad (3)$$

$$\mathcal{J}_{\mathcal{L}_\alpha^c} = \sum_{c' \neq c}^{N_M} \sum_{\beta \neq \alpha}^{N_I} (p_\beta^{c'} - \frac{1}{N_I} \sum_{\gamma=1}^{N_I} p_\gamma^{c'}) \mathcal{L}_{\alpha\beta}^{cc'} \quad (4)$$

$$S(\mathcal{F}, \mathcal{F}') = \frac{|\mathcal{F}|^T |\mathcal{F}'|}{\|\mathcal{F}\| \|\mathcal{F}'\|} \quad (5)$$

Handcrafted lateral connections. In former systems and publications [5,7,6,8] lateral connections were handcrafted. Neighboring control units of neighboring control columns were connected having a certain weight, all other weights are set to zero. Figure 2 shows the structure of these connections for one receiving control unit. These connections are drawn for all control units in all control columns. Here we want to learn these connections and do not create them by hand. But we expect the system to learn lateral connections that look similar.

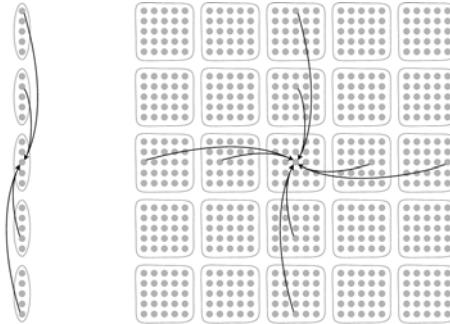


Fig. 2. Examples for a lateral connection pattern in 1D and 2D: Each control unit receives input from neighbored control units of neighbored control columns

4 Learning Lateral Connections

Instead of hand-crafting lateral connections we want the system to learn them based on statistical properties of the neighborhood of similar feature-vectors. Therefore we generalize learning in a single column [9] to learn in large scale networks of columns. Also note that the learning rules are different to the learning in [18]. The learning is described by the following set of equations:

$$if \quad \nu < \nu_{\max} \quad \text{and} \quad P > \chi \quad (6)$$

$$\Delta \mathcal{L}_{\alpha\beta}^{cc'} = \epsilon p_\alpha^c (S_\alpha^c S_{\beta}^{c'} - Z_{\alpha\beta}^c \mathcal{L}^{cc'}) \quad \text{with} \quad (7)$$

$$S_\alpha^c = S(\mathcal{F}_c^M, \mathcal{F}_\alpha^I) \quad \text{and} \quad (8)$$

$$S_{\beta}^{c'} = S(\mathcal{F}_{c'}^M, \mathcal{F}_\beta^I) \quad (9)$$

$$Z_{\alpha\beta}^c = S_\alpha^c \sum_{c'}^{N_M} S(\mathcal{F}_{c'}^M, \mathcal{F}_\beta^I) \quad (10)$$

$$P = \sum_c^{N_M} \sum_\alpha^{N_I} p_\alpha^c \quad (11)$$

The learning takes place in a window of the ν -cycle defined by the conditions $\nu < \nu_{\max}$ and $P > \chi$. The threshold χ of the sum of the activity of all control units of all control columns P defines the lower bound in terms of ν because

of increasing inhibition ν the sum of activity P is decreasing and at a certain value of ν getting smaller than χ . The upper bound is directly set by ν_{\max} . The actual change of a fiber from a control unit β of column c' to a control unit α of column c is calculated by a growth term and a decay-term. The growth is given by the product of the similarity S_α^c and the similarity of related model- and input-features $S_\beta^{c'}$. So the connection strength grows if both control units (p_α^c and $p_\beta^{c'}$) receive strong input from similarity between model and input. The decay is represented by the product of the similarity S_α^c with the sum of all similarities of the source-column c' . The change of the connection strength is proportional to the learning rate ϵ and the activity of the receiving control unit p_α^c . Let us, again, have a look at the window of learning defined by χ and ν_{\max} . Both bounds are dynamically balanced by (12) and (13), where λ_χ and λ_ν are parameters to tune the speed for changes of χ and ν_{\max} . While χ starts at a very low level and increases over time converging to a certain limit, ν_{\max} is balanced to have a value where only few control units are active. In the case of well matching correspondences the decision is taken early, such that the system most probably strengthens useful connections. For bad matching correspondences the system often needs larger inhibition to converge, so the state where k units are active lasts longer and the system learns from more than one possible solution for the match.

$$\Delta\chi = -\lambda_\chi(\chi - a_\chi P) \quad (12)$$

$$\Delta\nu_{\max} = -\lambda_\nu(a_\nu - \frac{P}{N_M}) \quad (13)$$

Learning from artificial data. In order to study the system's learning capabilities we, in a first experiment, generate random values for the feature vector entries of the model-grid instead of computing them from Gabor filters. Since there are no lateral connections in the beginning (their weight is zero) the model is represented by a bag of features. For each cycle in the simulation of the network we generate the grid of feature for the input in the following way: we copy the model-features arranged in a grid to a random position in the input, which itself is arranged as a larger grid. All remaining features of the input-grid are set to a random selected feature vector of the model-grid. This emulates ambiguous features of object and background in natural images. To distinguish between object features and background features the system has to learn to integrate feature similarity and similarity of feature arrangements. To measure the recognition rate we simulate the system with the inhibition ν starting at a low value ν_{start} and increasing beyond the end of learning ν_{\max} to a value ν_{end} where all control columns have only one active remaining control unit. Since we have generated the data we have ground truth knowledge and can count the columns where the correct control unit is active in comparison to the one, where the wrong ones remained active. Figure 5(a) shows the increasing recognition rate during learning, where learning is enabled in a later cycle. At the beginning of the simulation the recognition rate is at the level of pure chance, that the system finds the features within the arrangement. After learning is enabled, the

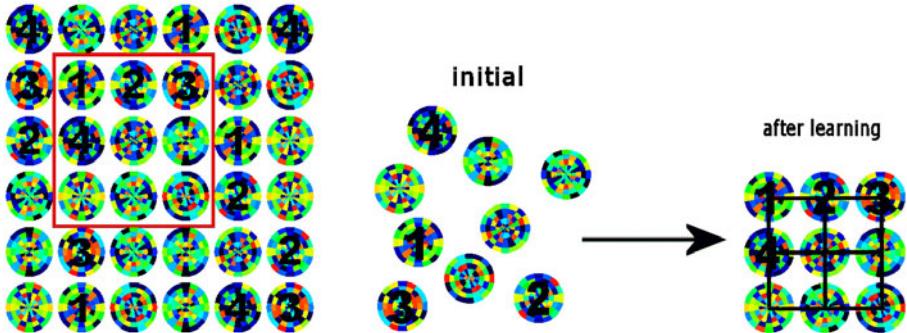


Fig. 3. Setup for learning with artificial data: we generate model feature vectors and copy them as a whole arrangement to a random position in the input-grid. All remaining features of the input-grid are set to a randomly selected feature vector of the model-grid. In this way we start with a bag of features model, which during learning evolves to a model with spacial feature relations.

recognition rate raises relatively quickly to the optimum. In other simulations we applied additive noise to the input features and observed slower learning curves not necessary reaching the optimum but still heading to a high recognition rate.

Using Gabor features from natural images. Since we now know that the system is able to solve the given task in an artificial setup we want to do the next step and present input gathered from natural images. We want to know if the system behaves the same for input from Gabor preprocessing instead of randomly generated values. For this purpose we find a setup similar to the artificial one but use Gabor preprocessing on real images. We divide a natural

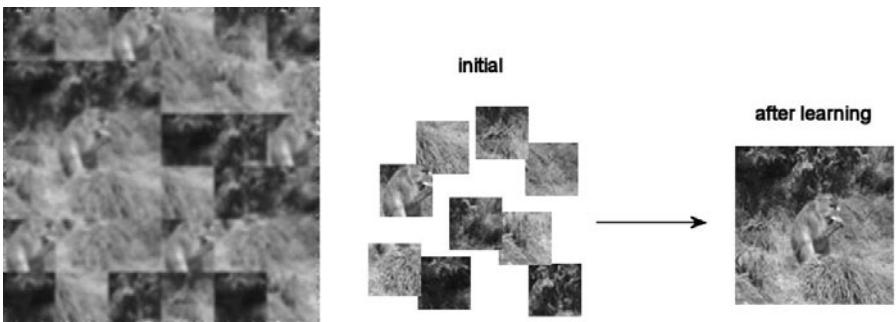


Fig. 4. Setup for more natural data: similar to the previous setup we generate an input image from a complete model image and permuted model feature vectors. In this case we also apply gaussian blurring to the tile composed input image in order to increase the difficulty of this task. In the beginning of the simulation the model features do not have any spatial relationship. Similar to the setup with artificial features, we expect the system to learn the correct spatial relations of the tiles.

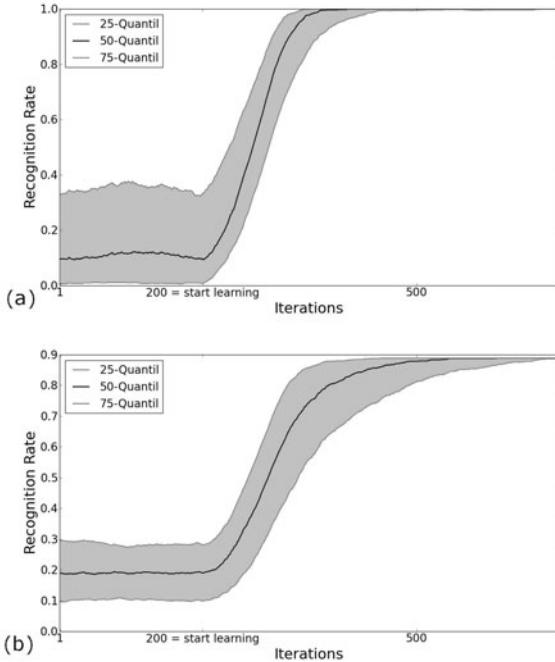


Fig. 5. The learning curves of the simulations show that the system is able to learn to integrate feature similarity and the similarity of feature arrangement. In order to show the recognition rate in the initial state with all lateral connections at zero we start the simulations without learning and enable learning after 200 iterations. As the figures show, learning nearly immediately increases the recognition rate until it reaches the optimum in the case of artificial data (a) and a quite high recognition rate in the case of features from blurred natural images (b). Without the gaussian blurring the system also reaches the optimum as we observed in other simulations.

image into tiles and compose the input from these tiles. The tile-composed image then consists of a section of correctly arranged tiles showing the original image and the same tiles at random positions serving as background clutter (see figure 4). By choosing the appropriate tile size (32×32 pixels) we avoid getting artefacts in the Gabor features generated by edges between tiles (the Gabor filter has an effective range of approximately 10 pixels). The input grid then consists of properly arranged tiles and random selected tiles like in the artificial setup. To make it still more difficult for the system we blurred (gaussian) the tile-composed input image before extracting Gabor features. Figure 5(b) shows the learning curve similar to the results of the artificial setup.

5 Discussion

Based on former work on column based networks we introduced a learning dynamics adapted from learning in a single column (see [9]). We showed that using

this dynamics the column network is able to learn spatial arrangements of features extracted from natural images. In contrast to the learning dynamics used in [18] we do not have to change any parameters if we use different numbers of cycles because of the self-balancing ability of our learning rule. The dynamically recalculated ν -window is the key to the stability of the learning-mechanism. This balancing has proven to robustly increase object detection rates during learning in artificial as well as more natural settings. The training data was chosen to be particularly challenging by using backgrounds containing similar or the same features as the object itself. As demonstrated, the system initially starts with a bag-of-features approach which gives very low recognition rates. During training, the system learns the typical feature arrangement of the object, however, and increases recognition rates to close to 100%. The system does hereby not use any prior knowledge about the feature arrangement. If we, for instance, trained on one dimensional input as, e.g., displayed in Fig. II one dimensional feature arrangements and one dimensional translations were learned. In principle, the setting allows to learn any arrangements and translation in any dimensionality. Also other forms of transformations can be learned from data, including mirroring, scaling, or rotation (compare [19] who incorporate these transformations without learning). Limitations of the features used have to be considered, however (e.g., Gabor-features are not rotation invariant). The most severe drawback of the presented system is so far the large demand of computational resources. By constraining the lateral connectivity from all-to-all to lower connection densities and by employing parallel simulation techniques we currently increase the system's efficiency. In future larger-scale applications we can thus use Gabor grids with more nodes for input and model, which will allow for applications to increasingly realistic application scenarios.

Acknowledgement. We thank Jörg Bornschein and Gervasio Puertas for technical help with code optimization and parallelization. Furthermore, we gratefully acknowledge funding by the German Federal Ministry of Education and Research (BMBF) in the project 01GQ0840 (BFNT Frankfurt) and funding by the German Research Foundation (DFG) in the project LU 1196/4-1.

References

1. Mel, B.W.: Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural. Comp.* 9, 777–804 (1997)
2. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. *Nature Neuroscience* 211(11), 1019–1025 (1999)
3. Hinton, G., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. *Neural Computation* 18, 1527–1554 (2006)
4. Friston, K.: Hierarchical models in the brain. *PLoS Computational Biology* 4(11), e10000211 (2008)
5. Olshausen, B.A., Anderson, C.H., Essen, D.C.V.: A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience* 13(11), 4700–4719 (1993)

6. Wiskott, L., von der Malsburg, C.: Face recognition by dynamic link matching. In: Sirosh, J., Miikkulainen, R., Choe, Y. (eds.) *Lateral Interactions in the Cortex: Structure and Function* (1995), ISBN 0-9647060-0-8
7. Lücke, J., Keck, C., von der Malsburg, C.: Rapid convergence to feature layer correspondences. *Neural Computation* 20(10), 2441–2463 (2008)
8. Wolfrum, P., Wolff, C., Lücke, J., von der Malsburg, C.: A recurrent dynamic model for correspondence-based face recognition. *Journal of Vision* 8(7), 1–18 (2008)
9. Lücke, J.: Receptive field self-organization in a model of the fine-structure in V1 cortical columns. *Neural Computation* 21(10), 2805–2845 (2009)
10. Hinton, G.E.: A Parallel Computation that Assigns Canonical Object-Based Frames of Reference. In: Proc. IJCAI, pp. 683–685 (1981)
11. Lücke, J., Bouecke, J.D.: Dynamics of cortical columns – self-organization of receptive fields. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) *ICANN 2005*. LNCS, vol. 3696, pp. 31–37. Springer, Heidelberg (2005)
12. Lücke, J., von der Malsburg, C.: Rapid Processing and Unsupervised Learning in a Model of the Cortical Macrocolumn. *Neural Computation* 16, 501–533 (2004)
13. Yoshimura, Y., Dantzker, J.L.M., Callaway, E.M.: Excitatory cortical neurons form fine-scale functional networks. *Nature* 433, 868–873 (2005)
14. Yuille, A.L., Geiger, D.: Winner-take-all networks. In: Arbib, M.A. (ed.) *The handbook of brain theory and neural networks*, pp. 1228–1231. MIT Press, Cambridge (2003)
15. Lücke, J., Sahani, M.: Maximal Causes for Non-linear Component Extraction. *Journal of Machine Learning Research* 9, 1227–1267 (2008)
16. Ringach, D.L.: Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of Neurophysiology* 88, 455–463 (2002)
17. Zhu, J., von der Malsburg, C.: Maplets for correspondence-based object recognition. *Neural Networks* 17(8-9), 1311–1326 (2004)
18. Bouecke, J.D., Lücke, J.: Learning of neural information routing for correspondence finding. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) *ICANN 2009*. LNCS, vol. 5769, pp. 557–566. Springer, Heidelberg (2009)
19. Sato, Y.D., Jitsev, J., Malsburg, C.: A visual object recognition system invariant to scale and rotation. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) *ICANN 2008*, Part I. LNCS, vol. 5163, pp. 991–1000. Springer, Heidelberg (2008)

Computational Properties of Probabilistic Neural Networks

Jiří Grim and Jan Hora

Institute of Information Theory and Automation
P.O. BOX 18, 18208 PRAGUE 8, Czech Republic
{grim,hora}@utia.cas.cz

Abstract. We discuss the problem of overfitting of probabilistic neural networks in the framework of statistical pattern recognition. The probabilistic approach to neural networks provides a statistically justified subspace method of classification. The underlying structural mixture model includes binary structural parameters and can be optimized by EM algorithm in full generality. Formally, the structural model reduces the number of parameters included and therefore the structural mixtures become less complex and less prone to overfitting. We illustrate how recognition accuracy and the effect of overfitting is influenced by mixture complexity and by the size of training data set.

Keywords: Probabilistic neural networks, Statistical pattern recognition, Subspace approach, Overfitting reduction.

1 Introduction

One of the most serious problems in training neural networks is the overfitting. In practical situations the real-life data sets are usually not very large and, moreover, they have to be partitioned into training- and test subsets to enable independent evaluation. Typically neural networks adapt satisfactorily to training data but the test set performance is usually worse. The bad "generalizing property" is caused by the "overfitting" of neural network parameters to the training data. Even slight differences between the statistical properties of the training- and test data may "spoil" the resulting independent evaluation.

Essentially, each of the two sets may be insufficiently "representative" with similar unhappy consequences. In order to avoid random influence of the data set partitioning one can use different cross-validation techniques. The repeated estimation of the neural network parameters and evaluation of its performance for differently chosen training- and test sets may be tedious, but it is the way to utilize the statistical information contained in data optimally. Nevertheless, even the result of a careful cross-validation can be biased if the source data set is insufficiently representative as a whole. Thus, e.g., a large portion of untypical samples (outliers) could negatively influence both the training- and test phase. In practical situation we usually assume the available data to be representative but, for obvious reasons, there is no possibility to verify it. For this reason the proper choice of the classifier and reduction of the risk of overfitting is of importance.

Generally, the risk of overfitting is known to increase with the problem dimension and model complexity and with the decreasing number of training samples. However, there is no commonly applicable rule to balance all these problem aspects. In literature we can find usually different recommendations and approaches to avoid overfitting in practical situations. In connection with neural networks the most common solution to avoid overfitting is to stop training [9]. Yinyiu et.al. [14] propose stopping rule for approximating neural networks based on signal-to-noise ratio. A related subject is the optimal complexity of classifiers. Thus, e.g., selective pruning has been proposed to increase the predictive accuracy of decision trees [10]. In this sense "under-computing" is also a way to avoid overfitting [2]. A general analysis of all the related questions is difficult since it is highly data dependent and classifier specific. For these and other reasons the problem of overfitting is rarely subject of theoretical papers.

In this paper we discuss the computational properties of probabilistic neural networks (PNN) recently developed in a series of papers (cf. [4] - [7]). Unlike the original concept of Specht [12] we use distribution mixtures of product components as a theoretical background. A special feature of the considered PNN is a structural mixture model including binary structural parameters which can be optimized by EM algorithm in full generality (cf. [4], [7]). By using structural mixtures the Bayes probabilities become proportional to weighted sums of component functions defined on different subspaces. The resulting subspace approach is statistically justified and can be viewed as a more general alternative of different feature selection methods. In this way we have a statistically correct tool to model incompletely interconnected neural networks. Both the structure and the training of PNN have a plausible biological interpretation with the mixture components playing the role of formal neurons [6].

Formally, the subspace approach based on the structural mixture model reduces the number of model parameters and therefore the structural mixtures become less prone to overfitting. In the following we analyze the tendency of PNN to overfitting in the framework of statistical pattern recognition. In order to illustrate different aspects of the problem we train PNN to recognize two classes of artificial chess-board patterns arising by random moves of the chess pieces rook and knight respectively. The underlying classification problem is statistically nontrivial with the possibility to generate arbitrary large data sets. We compare the structural mixture models with the complete multivariate Bernoulli mixtures and show, how the recognition accuracy is influenced by the mixture complexity and by the size of training data sets.

The paper is organized as follows. In Sec.2 we summarize basic features of PNN. Sec.3 describes the generation of training- and test data sets and the computational details of EM algorithm. In Sec. 4 we describe the classification performance of the fully interconnected PNN in dependence on model complexity and the size of training sets. In Sec. 5 we compare the tables of Sec. 4. with the analogous results obtained with the incomplete structural mixture model. Finally the results are summarized in the Conclusion.

2 Subspace Approach to Statistical Pattern Recognition

Let us consider the problem of statistical recognition of binary data vectors

$$\mathbf{x} = (x_1, x_2, \dots, x_N) \in \mathcal{X}, \quad \mathcal{X} = \{0, 1\}^N$$

to be classified according to a finite set of classes $\Omega = \{\omega_1, \dots, \omega_K\}$. Assuming the final decision-making based on Bayes formula

$$p(\omega|\mathbf{x}) = \frac{P(\mathbf{x}|\omega)p(\omega)}{P(\mathbf{x})}, \quad P(\mathbf{x}) = \sum_{\omega \in \Omega} P(\mathbf{x}|\omega)p(\omega), \quad \mathbf{x} \in \mathcal{X}, \quad (1)$$

we approximate the class-conditional distributions $P(\mathbf{x}|\omega)$ by finite mixtures of product components

$$P(\mathbf{x}|\omega) = \sum_{m \in \mathcal{M}_\omega} F(\mathbf{x}|m)f(m) = \sum_{m \in \mathcal{M}_\omega} f(m) \prod_{n \in \mathcal{N}} f_n(x_n|m), \quad \sum_{m \in \mathcal{M}_\omega} f(m) = 1. \quad (2)$$

Here $f(m) \geq 0$ are probabilistic weights, $F(\mathbf{x}|m)$ denote product distributions, \mathcal{M}_ω are the component index sets of different classes and \mathcal{N} is the index set of variables. In case of binary data vectors we assume the components $F(\mathbf{x}|m)$ to be multivariate Bernoulli distributions, i.e. we have

$$f_n(x_n|m) = \theta_{mn}^{x_n}(1 - \theta_{mn})^{1-x_n}, \quad n \in \mathcal{N}, \quad \mathcal{N} = \{1, \dots, N\}, \quad (3)$$

$$F(\mathbf{x}|m) = \prod_{n \in \mathcal{N}} f_n(x_n|m) = \prod_{n \in \mathcal{N}} \theta_{mn}^{x_n}(1 - \theta_{mn})^{1-x_n}, \quad m \in \mathcal{M}_\omega. \quad (4)$$

The basic idea of PNN is to assume the component distributions in Eq. (2) in the following "structural" form

$$F(\mathbf{x}|m) = F(\mathbf{x}|0)G(\mathbf{x}|m, \phi_m), \quad m \in \mathcal{M}_\omega \quad (5)$$

where $F(\mathbf{x}|0)$ is a "background" probability distribution usually defined as a fixed product of global marginals

$$F(\mathbf{x}|0) = \prod_{n \in \mathcal{N}} f_n(x_n|0) = \prod_{n \in \mathcal{N}} \theta_{0n}^{x_n}(1 - \theta_{0n})^{1-x_n}, \quad (\theta_{0n} = \mathcal{P}\{x_n = 1\}) \quad (6)$$

and the component functions $G(\mathbf{x}|m, \phi_m)$ include additional binary structural parameters $\phi_{mn} \in \{0, 1\}$

$$G(\mathbf{x}|m, \phi_m) = \prod_{n \in \mathcal{N}} \left[\frac{f_n(x_n|m)}{f_n(x_n|0)} \right]^{\phi_{mn}} = \prod_{n \in \mathcal{N}} \left[\left(\frac{\theta_{mn}}{\theta_{0n}} \right)^{x_n} \left(\frac{1 - \theta_{mn}}{1 - \theta_{0n}} \right)^{1-x_n} \right]^{\phi_{mn}}. \quad (7)$$

Note that formally the structural component (5) is again the multivariate Bernoulli distribution (4) with the only difference, that for $\phi_{mn} = 1$ the corresponding estimated parameter θ_{mn} is replaced by the fixed background parameter θ_{0n} :

$$F(\mathbf{x}|m) = \prod_{n \in \mathcal{N}} [\theta_{mn}^{x_n} (1 - \theta_{mn})^{1-x_n}]^{\phi_{mn}} [\theta_{0n}^{x_n} (1 - \theta_{0n})^{1-x_n}]^{1-\phi_{mn}}. \quad (8)$$

An important feature of PNN is the possibility to optimize the mixture parameters $f(m)$, θ_{mn} , and the structural parameters ϕ_{mn} simultaneously by means of the iterative EM algorithm (cf. [11][18][3]). In particular, given a training set of independent observations from the class $\omega \in \Omega$

$$\mathcal{S}_\omega = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(K_\omega)}\}, \quad \mathbf{x}^{(k)} \in \mathcal{X},$$

we compute the maximum-likelihood estimates of the mixture parameters by maximizing the log-likelihood function

$$L = \frac{1}{|\mathcal{S}_\omega|} \sum_{x \in \mathcal{S}_\omega} \log P(\mathbf{x}|\omega) = \frac{1}{|\mathcal{S}_\omega|} \sum_{x \in \mathcal{S}_\omega} \log \left[\sum_{m \in \mathcal{M}_\omega} F(\mathbf{x}|0) G(\mathbf{x}|m, \phi_m) f(m) \right]. \quad (9)$$

For this purpose the EM algorithm can be modified as follows (cf. [3][4]):

$$f(m|\mathbf{x}) = \frac{G(\mathbf{x}|m, \phi_m) f(m)}{\sum_{j \in \mathcal{M}_\omega} G(\mathbf{x}|j, \phi_j) f(j)}, \quad m \in \mathcal{M}_\omega, \quad \mathbf{x} \in \mathcal{S}_\omega, \quad (10)$$

$$f'(m) = \frac{1}{|\mathcal{S}_\omega|} \sum_{x \in \mathcal{S}_\omega} f(m|\mathbf{x}), \quad \theta'_{mn} = \frac{1}{|\mathcal{S}_\omega| f'(m)} \sum_{x \in \mathcal{S}_\omega} x_n f(m|\mathbf{x}), \quad n \in \mathcal{N} \quad (11)$$

$$\gamma'_{mn} = f'(m) \left[\theta'_{mn} \log \frac{\theta'_{mn}}{\theta_{0n}} + (1 - \theta'_{mn}) \log \frac{(1 - \theta'_{mn})}{(1 - \theta_{0n})} \right], \quad (12)$$

$$\phi'_{mn} = \begin{cases} 1, & \gamma'_{mn} \in \Gamma', \\ 0, & \gamma'_{mn} \notin \Gamma', \end{cases}, \quad \Gamma' \subset \{\gamma'_{mn}\}_{m \in \mathcal{M}_\omega, n \in \mathcal{N}}, \quad |\Gamma'| = r. \quad (13)$$

Here $f'(m)$, θ'_{mn} , and ϕ'_{mn} are the new iteration values of mixture parameters and Γ' is the set of a given number of the highest quantities γ'_{mn} . The iterative equations (10)-(13) generate a nondecreasing sequence $\{L^{(t)}\}_0^\infty$ converging to a possibly local maximum of the log-likelihood function (9).

Let us note that Eq. (12) can be expressed by means of Kullback-Leibler discrimination information (see e.g. [13]) between the conditional component-specific distribution $f'_n(\cdot|m) = \{\theta'_{mn}, 1 - \theta'_{mn}\}$ and the corresponding univariate “background” distribution $f_n(\cdot|0) = \{\theta_{0n}, 1 - \theta_{0n}\}$:

$$\gamma'_{mn} = f'(m) I(f'_n(\cdot|m) || f_n(\cdot|0)) \quad (14)$$

In this sense the r -tuple of the most informative conditional distributions $f'_n(\cdot|m)$ is included in the structural mixture model at each iteration.

The main advantage of the structural mixture model is the possibility to cancel the background probability density $F(\mathbf{x}|0)$ in the Bayes formula since then the decision making may be confined only to “relevant” variables. In particular, denoting $w_m = p(\omega)f(m)$ we can write

$$p(\omega|\mathbf{x}) = \frac{\sum_{m \in \mathcal{M}_\omega} G(\mathbf{x}|m, \phi_m) w_m}{\sum_{j \in \mathcal{M}} G(\mathbf{x}|j, \phi_j) w_j} \approx \sum_{m \in \mathcal{M}_\omega} G(\mathbf{x}|m, \phi_m) w_m, \quad \mathcal{M} = \bigcup_{\omega \in \Omega} \mathcal{M}_\omega. \quad (15)$$

Consequently the posterior probability $p(\omega|\mathbf{x})$ becomes proportional to a weighted sum of the component functions $G(\mathbf{x}|m, \phi_m)$, each of which can be defined on a different subspace. In other words the input connections of a neuron can be confined to an arbitrary subset of input variables. In this sense the structural mixture model represents a statistically justified subspace approach to Bayesian decision-making which is directly applicable to the input space without any preliminary feature selection or dimensionality reduction [34]. The structural mixture model can be viewed as a strong alternative to the standard feature selection techniques whenever the measuring costs of the input variables are negligible.

3 Artificial Classification Problem

In order to analyze the different aspects of overfitting experimentally we need sufficiently large data sets. As the only reasonable way to satisfy this condition is to generate artificial data, we consider in the following the popular problem of recognition of chess-board patterns generated by random moving of the chess pieces rook and knight. The problem is statistically nontrivial, it is difficult to extract a small subset of informative features without essential information loss. Unlike the original version (cf. [5]) we have used 16×16 chess board instead of 8×8 since the resulting dimension $N = 256$ should emphasize the role data set size. The chess-board patterns were generated randomly with the initial position randomly chosen. We started with all chess-board fields marked by the binary value 0, each position of the chess piece was marked by 1. Finally the chess-board pattern is rewritten in the form of a 256-dimensional binary vector.

In case of rook-made patterns (class 1), we generated first the initial position by means of a random integer from the interval $\langle 1, 256 \rangle$ and then we chose one of the two possible directions (vertical, horizontal) with equal probability. In the second step we chose one of the 15 equiprobable positions in the specified direction. Possible returns to the previous positions were allowed. Random moves have been repeated until 10 different positions of the chess piece were marked.

Similarly, in case of the knight-made patterns (class 2), we generated again the initial position randomly. To make a random move we identified all the fields (maximally 8) which can be achieved by the chess piece knight from its current position (maximally 8) and then the next position of the knight has been chosen randomly. Again, possible returns to the previous position were not excluded and we repeated the random moves until 10 different labeled positions of the

knight. In this way we obtained in both classes 256-dimensional binary vectors containing always 10 ones a 246 zeros.

The number of labeled positions on the chess board is rather relevant pattern parameter from the point of view of problem complexity. Surprisingly, the recognition of the chess board patterns seems to simplify with the increasing number of generated positions, except for nearly completely covered chess board. By the present choice of 10 different positions we have tried to make the recognition more difficult.

We recall that the classes may overlap because there are many patterns which can be obtained both by moving knight and rook. Also, the dimension is relatively high ($N = 256$). The patterns have strongly statistical nature without any obvious deterministic properties. By comparing the marginal probabilities of both classes we can see nearly uniformly distributed values, only the knight achieves the margins and corners of the chess board with slightly lower probabilities. Consequently, there is no simple way to extract a small informative subset of features without essential information loss.

In order to solve the two-class recognition problem by means of Bayes formula (15) we estimate the respective class-conditional distribution mixtures of the form (2) separately for each class by using EM algorithm (10)-(13). Instead of Eq. (13) we use computationally more efficient thresholding. In particular, we have defined the structural parameters $\phi'_{mn} = 1$ as follows:

$$\phi'_{mn} = \begin{cases} 1, & \gamma'_{mn} \geq C_0 \gamma_0, \\ 0, & \gamma'_{mn} < C_0 \gamma_0, \end{cases}, \quad \gamma_0 = \frac{1}{N|\mathcal{M}_\omega|} \sum_{m \in \mathcal{M}_\omega} \sum_{n \in \mathcal{N}} \gamma'_{mn}. \quad (16)$$

In Eq. (16) the threshold value is related to the mean informativity γ_0 by a coefficient C_0 . In our experiments we have used $C_0 = 0.1$ since in this way only the less informative parameters are excluded from the evaluation of components. We assume that just these parameters are likely to be responsible for overfitting in the full mixture model. Note that, essentially, multivariate Bernoulli mixtures with many components may "overfit" by fitting the components to more specific patterns or even to outliers. In such a case the univariate distributions $f_n(x_n|m)$ are degraded to delta-functions and the corresponding component weight decreases to $f'(m) = 1/|\mathcal{S}_\omega|$. Consequently, the informativity γ'_{mn} of the related parameters θ'_{mn} decreases because of the low component weight $f'(m)$ in Eq. (14). In this way the structural mixture model can reduce overfitting by excluding highly specific components.

In order to simplify the comparison with the full mixture model, we have used twice more mixture components in structural mixtures. As a result the complexity of the structural model was comparable with the related full mixture model in terms of the final number of parameters. In the experiments we have used first the complete model with all parameters included (cf. Sec. 4, Table 1) and then the results have been compared with the structural mixture model (Sec. 5, Table 2).

In several computational experiments (cf. Sec. 4, Table 1, Sec. 5, Table 2) we have varied both the number of mixture components, i.e. the complexity of

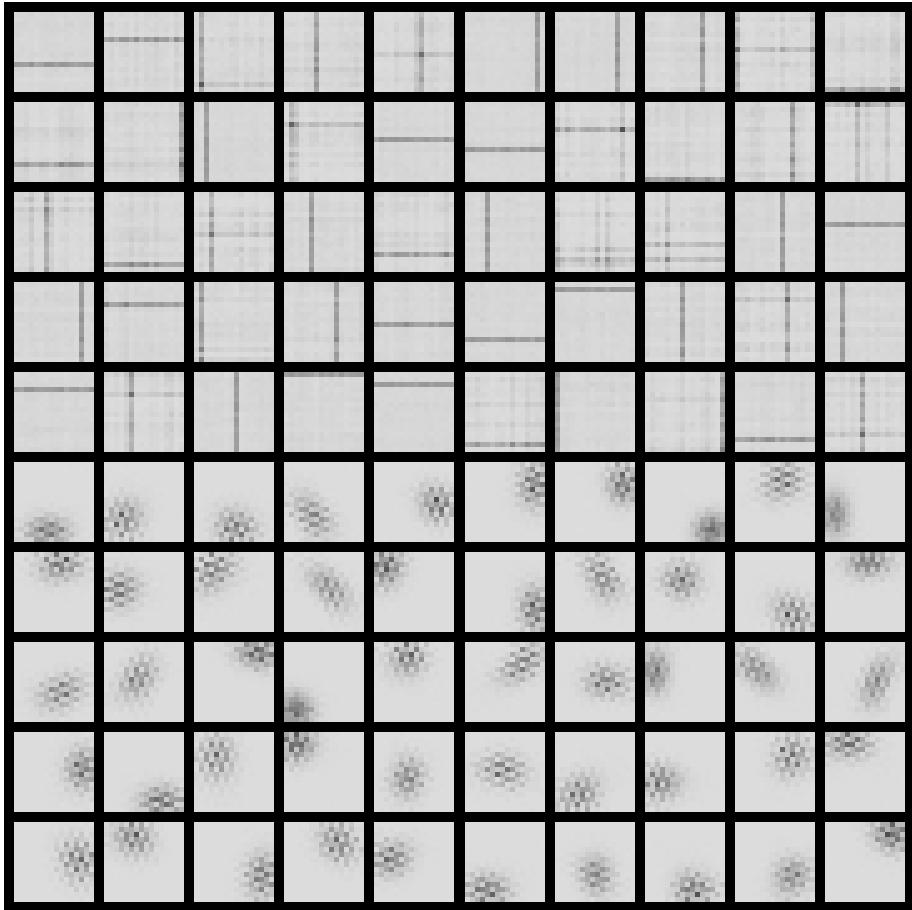


Fig. 1. Component parameters θ_{mn} in chess-board arrangement corresponding to the typical variants of samples ($|\mathcal{M}_\omega| = 50$). The parameters θ_{nm} are displayed by grey-levels inversely (0 = white, 1 = black). The different "style" of patterns of both classes is well apparent.

the mixture model, ($|\mathcal{M}_\omega| = 1, 2, 5, 10, 20, 50, 100, 200, 500$) and the size of the training set ($|\mathcal{S}_\omega| = 10^3, 10^4, 10^5$). For the sake of independent testing we have used one pair of sufficiently large sets (2×10^5 patterns for each class). Obviously this approach is not very realistic since in practical situation a large test set is usually not available. However, in this way the test set performance in our experiments is verified with a high level of reliability and we avoid an additional source of noise. In all computations the initial mixture parameters θ_{nm} have been chosen randomly with the uniform weights $f(m) = 1/|\mathcal{M}_\omega|$. Certainly, there is some random influence of the starting point (especially in case of the small sample size) but in repeated computations with different starting points we have obtained comparable results. The EM-iterations have been stopped

Table 1. Results of numerical experiments with full multivariate Bernoulli mixtures

M	K	1 000	200 000	10 000	200 000	100 000	200 000
2	512	34.70 %	41.56 %	39.59 %	40.38 %	39.90 %	40.02 %
4	1024	13.10 %	15.83 %	16.54 %	16.65 %	16.42 %	16.48 %
10	2560	1.65 %	7.72 %	6.60 %	7.00 %	6.49 %	6.60 %
20	5120	0.95 %	9.21 %	5.40 %	5.90 %	4.04 %	4.34 %
40	10240	0.15 %	8.76 %	3.91 %	4.90 %	2.73 %	2.89 %
100	25600	0.00 %	9.35 %	2.01 %	4.54 %	1.37 %	1.90 %
200	51200	0.00 %	11.02 %	1.22 %	5.57 %	0.84 %	1.68 %
400	102400	0.00 %	15.40 %	0.69 %	8.35 %	0.45 %	1.92 %
1000	256000	0.00 %	17.77 %	0.20 %	14.66 %	0.14 %	3.76 %

when the relative increment of the likelihood function $(Q' - Q)/Q$ was less than 0.0001. In all experiments the convergence of EM algorithm has been achieved in less than 40 iterations. The resulting estimates of the conditional distributions $P(\mathbf{x}|\omega_1), P(\mathbf{x}|\omega_2)$ have been used to perform the Bayesian classification both of the training- and independent test patterns according to maximum *a posteriori* probability (cf. (II), (I5)).

Figure 1 graphically displays the parameters of the estimated full mixture components ($|\mathcal{M}_\omega| = 50$) both for the class “knight” and “rook”. The parameters θ_{nm} are arranged to correspond to the underlying chess board, the parameter values are displayed by grey-levels inversely (0 = white, 1 = black). Note that the style of the component patterns suggests the nature of random moves.

4 Example - Full Mixture Model

In the first part of the numerical experiment we have used full mixture model including 256 parameters for each component. We estimated the class-conditional mixtures for the number of components increasing from $M=1$ to $M=500$ in each class. The total number of components ($2M$) for the respective solution is given in the first column of Table 1, the second column contains the corresponding total number of parameters increasing form $K=512$ to a relatively high value $K=256000$. For simplicity, we do not count component weights as parameters.

The class-conditional mixtures have been estimated for three differently large training sets ($|\mathcal{S}_\omega| = 10^3, 10^4, 10^5$). The corresponding recognition error computed on the training set is given in the columns 3, 5 and 7 respectively, the corresponding error estimated on the independent test set ($|\mathcal{S}_\omega| = 2 \times 10^5$) is given in the columns 4,6 and 8 respectively. It can be seen that even small training data set ($|\mathcal{S}_\omega| = 10^3$) is sufficient to estimate the underlying 256 parameters reasonably. The classification error on the test set is about 40% and it is comparable with the respective training-set error. Thus, in the second row, the effect of overfitting is rather small because of the underlying model simplicity. In the last row the complex mixtures of 500 components (1000 totally) strongly overfit by showing the training set error near zero. The related independent test set

Table 2. Results of numerical experiments with the structural mixture model

M	K	1 000	200 000	10 000	200 000	100 000	200 000
4	830	13.80 %	16.48 %	25.53 %	26.53 %	16.91 %	16.92 %
8	1650	6.45 %	9.97 %	10.96 %	11.32 %	7.28 %	7.29 %
20	3770	5.70 %	10.97 %	5.14 %	5.77 %	4.70 %	4.72 %
40	7010	8.65 %	13.63 %	4.20 %	4.73 %	3.29 %	3.32 %
80	13500	4.20 %	12.91 %	6.12 %	6.88 %	1.91 %	1.92 %
200	32700	0.25 %	11.46 %	3.36 %	4.76 %	1.83 %	1.85 %
400	65000	0.00 %	18.11 %	3.54 %	4.70 %	3.10 %	3.20 %
800	128100	0.00 %	18.50 %	3.88 %	4.82 %	5.42 %	5.45 %
2000	317200	0.00 %	18.75 %	2.84 %	6.39 %	2.71 %	2.73 %

error ($|\mathcal{S}_\omega| = 2 \times 10^5$) decreases from about 17.77% to 3.76% in the last column. Obviously the class-conditional mixtures including 128 thousand of parameters need more data to be reasonably estimated, otherwise the effect of overfitting is rather strong. Because of overfitting the recognition of training patterns is generally more accurate than that of independent test patterns. Expectedly, the training-set error decreases with the increasing complexity of class conditional mixtures because of increasingly tight adaptation to training data. However, because of overfitting, the training-set error would be a wrong criterion to choose model complexity. It can be seen that, in Table 1, the optimal number of components is different for different size of training set. In view of the independent test-set error we should use 5 components for each class-conditional mixture in case of training set $|\mathcal{S}_\omega| = 10^3$, (M=10, row 4), 50 components for the training set $|\mathcal{S}_\omega| = 10^4$, (M=100, row 7) and 100 components for $|\mathcal{S}_\omega| = 10^5$, (M=200, row 8).

5 Example - Structural Mixture Model

In the second part of the numerical experiments we recomputed the results of Table 1 by means of structural mixture model. By using a low threshold coefficient $C_0 = 0.1$ (cf. Eq. (16)) we exclude only parameters which are nearly superfluous in evaluation of mixture components. In order to compensate for the omitted parameters we have used twice more mixture components. In this way we have obtained a similar model complexity in the second column of Table 2.

At the first view the recognition accuracy in the second row is much better than in the Table 1. The differences illustrate the trivial advantage of the two-component mixtures in comparison with the single product components in Table 1. Expectedly, the structural mixture model is much less overfitting and therefore the differences between the respective training-set- and test-set errors are smaller (possibly except for the small training set $|\mathcal{S}_\omega| = 10^3$, column 3 and 4). In case of the two large training sets ($|\mathcal{S}_\omega| = 10^4, 10^5$) the overfitting effect is almost negligible. In the last rows the training set errors are greater than in Table 1 because the information criterion (14) reduces the fitting of components to

highly specific types of patterns. As it can be seen, the training-set error could be used to select the structural model complexity almost optimally. Thus we would choose M=200 components both for the training sets $|\mathcal{S}_\omega| = 10^4$ and 10^5 .

6 Concluding Remark

The complex multivariate Bernoulli mixtures with many components tend to overfit mainly by fitting the components to highly specific types of patterns or even to outliers. We assume that the "less informative" parameters are responsible for overfitting of the "full" multivariate Bernoulli mixture in Table 1. This mechanism of overfitting may be reduced by using structural mixture model since the fitting of components to outliers is disabled by means of informativity threshold. As it can be seen in Table 2 the resulting training set error of structural mixtures would enable almost correct choice of the model complexity.

Acknowledgement. Supported by the project GAČR No. 102/07/1594 of Czech Grant Agency and by the projects 2C06019 ZIMOLEZ and MŠMT 1M0572 DAR.

References

1. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Stat. Soc. B* 39, 1–38 (1977)
2. Dietterich, T.: Overfitting and undercomputing in machine learning. *ACM Computing Surveys* 27(3), 326–327 (1995)
3. Grim, J.: Multivariate statistical pattern recognition with nonreduced dimensionality. *Kybernetika* 22, 142–157 (1986)
4. Grim, J.: Information approach to structural optimization of probabilistic neural networks. In: Ferrer, L., et al. (eds.) *Proceedings of 4th System Science European Congress*, pp. 527–540. Soc. Espanola de Sistemas Generales, Valencia (1999)
5. Grim, J., Just, P., Pudil, P.: Strictly modular probabilistic neural networks for pattern recognition. *Neural Network World* 13, 599–615 (2003)
6. Grim, J.: Neuromorphic features of probabilistic neural networks. *Kybernetika* 43(6), 697–712 (2007)
7. Grim, J., Hora, J.: Iterative principles of recognition in probabilistic neural networks. *Neural Networks* 21(6), 838–846 (2008)
8. McLachlan, G.J., Peel, D.: *Finite Mixture Models*. John Wiley and Sons, New York (2000)
9. Sarle, W.S.: Stopped training and other remedies for overfitting. In: *Proceedings of the 27th Symposium on the Interface* (1995), <ftp://ftp.sas.com/pub/neural/inter95.ps.Z>
10. Schaffer, C.: Overfitting avoidance as Bias. *Machine Learning* 10(2), 153–178 (1993)
11. Schlesinger, M.I.: Relation between learning and self-learning in pattern recognition. *Kibernetika* (Kiev) (2), 81–88 (1968) (in Russian)
12. Specht, D.F.: Probabilistic neural networks for classification, mapping or associative memory. In: Proc. IEEE Int. Conf. on Neural Networks, vol. I, pp. 525–532 (1988)
13. Vajda, I.: *Theory of Statistical Inference and Information*. Kluwer, Boston (1992)
14. Yinyin, L., Starzyk, J.A., Zhu, Z.: Optimized Approximation Algorithm in Neural Networks Without Overfitting. *IEEE Tran. Neural Networks* 19, 983–995 (2008)

Local Minima of a Quadratic Binary Functional with a Quasi-Hebbian Connection Matrix

Yakov Karandashev, Boris Kryzhanovsky, and Leonid Litinskii

CONT SRISA RAS, Moscow, Vavilova St., 44/2, 119333, Russia

Yakov.Karandashev@phystech.edu, kryzhanov@mail.ru, litin@mail.ru
<http://www.niisi.ru/iont>

Abstract. The local minima of a quadratic functional depending on binary variables are discussed. An arbitrary connection matrix can be presented in the form of quasi-Hebbian expansion where each pattern is supplied with its own individual weight. For such matrices statistical physics methods allow one to derive an equation describing local minima of the functional. A model where only one weight differs from other ones is discussed in details. In this case the equation can be solved analytically. Obtained results are confirmed by computer simulations.

Keywords: Associative neural networks, energy landscape, statistical physics.

1 Introduction

Let us examine the problem of minimization of quadratic functional depending on N binary variables $S_i = \{\pm 1\}$, $i = 1, 2, \dots, N$:

$$E(\mathbf{S}) = -\frac{1}{N^2} \sum_{i,j=1}^N J_{ij} S_i S_j \xrightarrow{\text{s}} \min . \quad (1)$$

This problem arises in different scientific fields. The state of the system as a whole is given by N -dimensional vector $\mathbf{S} = (S_1, S_2, \dots, S_N)$ with binary coordinates S_i . These vectors will be called *configuration vectors* or simply *configurations*. The functional $E = E(\mathbf{S})$, which has to be minimized, will be called *the energy* of the state. The connection matrix $\mathbf{J} = (J_{ij})$ can be considered as a symmetric one since the substitution $J_{ij} \rightarrow (J_{ji} + J_{ij})/2$ does not change the value of E .

Statistical physics methods were efficiently used for analysis of the energy surface in the Hopfield model [1]-[2] with the Hebb connection matrix constructed from a set of M given patterns $\xi^\mu = (\xi_1^\mu, \xi_2^\mu, \dots, \xi_N^\mu)$, $\xi_i^\mu = \{\pm 1\}$, $\mu = 1, 2, \dots, M$. The result is as follows: if patterns ξ^μ are randomized and $M < 0.14 \cdot N$, in the immediate proximity of each pattern there is necessarily a local minimum of the functional (1). In other words, the neural network works as an associative memory [1].

In the paper [3] it was shown that any symmetric matrix can be presented as quasi-Hebbian expansion in uncorrelated configurations ξ^μ :

$$J_{ij} = \sum_{\mu=1}^M r_\mu \xi_i^\mu \xi_j^\mu. \quad (2)$$

In Eq. (2) weights r_μ are determined from the condition of non-correlatedness of configurations ξ^μ , and the number M is determined by the accuracy of approximation of the original matrix \mathbf{J} . The expression (2) is called the quasi-Hebbian representation since weights r_μ are different. We hope that this representation would allow one to use statistical physics methods for analysis of an arbitrary connection matrix. First obtained results are presented in this publication.

In Section 2 with the aid of statistical physics methods the principal equation for a connection matrix of the type (2) is derived. In Section 3 we analyze the case when only one weight differs from the others, which are identically equal: $r_1 \neq r_2 = r_3 = \dots = r_M = 1$. This model case can be analyzed analytically up to the end. It was found that a single weight that differs from 1 substantially affects the properties of local minima. Computer simulations confirm this result. Some conclusions and remarks are given in Section 4.

2 Principle Equation

Let $\mathbf{S} = (S_1, S_2, \dots, S_N)$, where $S_i = \pm 1$ define a state of the system. The energy $E(\mathbf{S})$ of the state can be presented in the form

$$E(\mathbf{S}) = - \sum_{\mu=1}^M r_\mu \left(m_\mu^2(\mathbf{S}) - \frac{1}{N} \right), \quad m_\mu(\mathbf{S}) = \frac{1}{N} \sum_{i=1}^N S_i \xi_i^\mu \quad (3)$$

where $m_\mu(\mathbf{S})$ is the overlap of the state \mathbf{S} with the pattern ξ^μ .

Statistical physics methods allow one to obtain equations for the overlap of a *local minimum* of the functional (1) with the patterns ξ^μ . Supposing the dimensionality of the problem to be very large ($N \gg 1$), let us set that the number of patterns M is proportional to N : $M = \alpha \cdot N$. Coefficient α is called the *load parameter*.

Repeating for the matrix (2) calculations performed in [1]-[2], we obtain the final equation for the local minimum in the vicinity of the pattern ξ^k :

$$\frac{1}{\alpha} = \frac{1}{\gamma^2} \cdot \frac{1}{M} \sum_{\mu \neq k}^M \frac{r_\mu^2}{(r_k \cdot \varphi - r_\mu)^2}, \quad (4)$$

where

$$\gamma = \sqrt{\frac{2}{\pi}} e^{-y^2} \quad \text{and} \quad \varphi = \frac{\sqrt{\pi}}{2} \frac{\operatorname{erf} y}{y} e^{y^2}. \quad (5)$$

The auxiliary variable y is positive, the function $\gamma = \gamma(y)$ decreases, and the function $\varphi = \varphi(y)$ increases monotonically from its minimal value $\varphi(0) = 1$. In what follows these functions are frequently used; for simplicity sometimes we omit their arguments, and use the notations γ and φ .

If y_0 is a solution of Eq. (4) we can calculate two main characteristics of the local minimum from the vicinity of ξ^k : its overlap m_k with ξ^k and $\sigma_k^2 = \sum_{\mu \neq k}^M r_\mu^2 m_\mu^2$ that is the weighted sum of squared overlaps of the local minimum with all patterns except the k -th one:

$$m_k = \operatorname{erf} y_0, \quad \sigma_k = \frac{r_k m_k}{\sqrt{2} y_0}. \quad (6)$$

An important role plays the determination of the critical value of the load parameter α_c for which the solution of Eq. (4) still exists, but for α that are larger than α_c there is no solution of Eq.(4). All characteristics corresponding to α_c are marked off with the same subscript, namely they are y_c , m_c and σ_c .

For the Hopfield model ($r_\mu \equiv 1$) Eq. (4) transforms in well-known equation

$$\alpha = \gamma^2(\varphi - 1)^2 \quad (7)$$

(see Eq. (2.74) in [2]). The critical value α_c is determined by the maximum value of the right-hand side of Eq. (7) and it is equal to $\alpha_c \approx 0.138$. This maximum has place at the point $y_c \approx 1.511$ that gives as the critical value of the overlap: $m_c \approx 0.967$. We see that even for the maximal value of the load parameter α_c , the local minimum is very close to the pattern. If the load parameter α is less than α_c , the overlap of the local minimum with the pattern even closer to 1. On the contrary, when the parameter α exceeds the critical value α_c , there is no solution of Eq. (7). As they say, “a breakdown” of the solution happens: the overlap decreases abruptly almost to zero value. In terms of statistical physics this means that in our system the first kind transition takes place. In terms of neural networks it means the associative memory destroying.

3 The Case of One Differing Pattern

Interesting is the case when all weights r_μ except one are equal to 1, and only one weight differs from other. Without loss of generality we can write

$$r_1 = \tau, \quad r_2 = r_3 = \dots = r_M = 1. \quad (8)$$

The first weight can be both larger and less than one.

It might seem that the difference from the classical Hopfield model has to be small: enormous number of patterns with the same weight takes part in the formation of the connection matrix and only one pattern provides a different contribution. Intuition

suggests that the influence of a single pattern with the individual weight τ has to be negligible small comparing with contribution of infinitely large number of patterns with the same weight $r_\mu = 1$. However, this is not the case: the unique weight τ can substantially affect the distribution of local minima. Let us examine separately what happens with local minimum in the vicinity of ξ^1 , and what happens with local minima near other patterns whose weights are equal to 1.

3.1 Pattern with an Individual Weight: $r_1 = \tau$.

For this pattern equation (4) has the form

$$\alpha = \gamma^2 (\tau \cdot \varphi - 1)^2. \quad (9)$$

If y is a solution of Eq. (9), the overlap of the local minimum with the first pattern is $m^{(1)} = \text{erf}(y)$. The superscript “(1)” emphasizes that we deal with the overlap of the local minimum with the pattern number 1.

The point of breakdown $y_c^{(1)}(\tau)$, where the right-hand side of Eq. (9) reaches its maximum, is the solution of the equation

$$\varphi(y) = 1 + \frac{2y^2}{\tau}. \quad (10)$$

After finding $y_c^{(1)}(\tau)$ the critical characteristics $m_c^{(1)} = \text{erf}(y_c^{(1)})$ and $\alpha_c^{(1)}$ can be calculated. It turns out that for $\tau > 1$ Eq. (10) has a nontrivial solution only if $\tau \leq 3$. When τ is larger than 3, this equation has only trivial solution: $\tau > 3 \Rightarrow y_c^{(1)}(\tau) \equiv 0$. This means that the overlap of the local minimum with the pattern vanishes: $m_c^{(1)} = \text{erf}(y_c^{(1)}) \equiv 0$. So, when $\tau > 3$ in the system the phase transition disappears. This is an unexpected result. We do not understand the reason why the phase transition disappears. (We have for $\tau \geq 3$: $\sigma_c^{(1)} = \tau \sqrt{\frac{2}{\pi}}$ and $\alpha_c^{(1)} = 2(\tau - 1)^2 / \pi$.)

In Fig.1a we present graphs of right-hand side of Eq. (9) for three different values of the weight coefficient: $\tau = 1$, $\tau = 2$ and $\tau = 3$. We see that increasing of the coefficient τ above 1 on the one hand is accompanied by an increase of the critical value $\alpha_c^{(1)}(\tau)$, and on the other hand it is accompanied by steady removal of the breakpoint $y_c^{(1)}(\tau)$ toward 0. As a result, when τ increases the overlap of the local minimum with the pattern in the critical point $m_c^{(1)}(\tau)$ decreases.

This behavior relates only to the critical values $y_c^{(1)}(\tau)$ and $m_c^{(1)}(\tau)$, i.e. to their values at the breakdown point. Absolutely other situation takes place if $\alpha < \alpha_c$. In Fig.1a show intersections between a straight line parallel to abscissa axis and the graphs representing the right-hand side of Eq. (9) for different τ . Abscissa values of each intersection are solutions of the equation. We see that when τ increases the

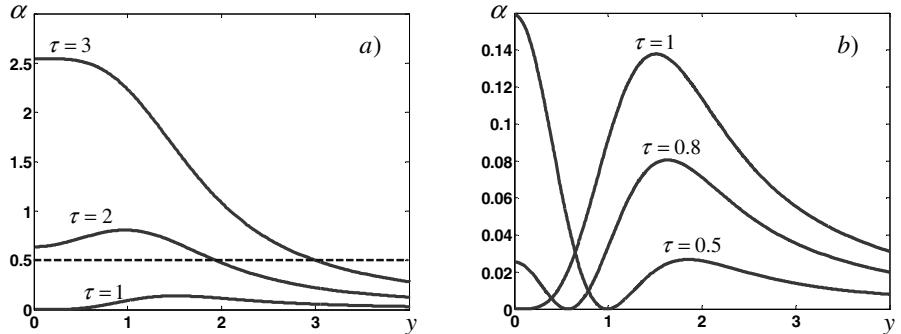


Fig. 1. The graphs of the right-hand side of Eq. (9) for different values of τ . a) $\tau \geq 1$; the dashed straight line intersects the graphs corresponding to different values of τ . b) $\tau \leq 1$; minima of each graph are equal to zero.

point of intersection shifts to the right, i.e. in the direction of its greater values. This means that when τ increases the overlap $m^{(1)}(\tau)$ of the local minimum with the pattern also increases. This behavior of the overlap is in agreement with the common sense: the greater the weight of the pattern τ , the greater its influence. Then the overlap of this pattern with the local minimum has to be greater.

Now let us examine the interval $[0,1]$ of the weight τ . Analysis of Eqs. (9), (10) shows that when τ decreases from 1 to 0 the maximum point $y_c^{(1)}(\tau)$ steadily shifts to the right, and the maximum value $\alpha_c^{(1)}$ steadily decreases (see the behavior of maxima of the curves in Fig.1b). When $\tau \leq 1$, the function $\tau \cdot \varphi(y) - 1$ in the point of minimum vanishes necessarily. The behavior of curves to the left from minima does not interest us, because these solutions of Eq. (9) correspond not to the minimum of the free energy, but to stationary points only [2].

3.2 Patterns with the Same Weight: $r_\mu = 1, \mu \geq 2$

Let us examine how the overlap of the local minimum with one of the patterns whose weight is equal to 1 depends on τ . Since all these patterns are equivalent, we choose the pattern with number “2”. With the superscript (2) we mark off the characteristics $m^{(2)}, y^{(2)}, \alpha^{(2)}$ that are interesting for us.

Now Eq. (4) has the form:

$$L(y) = \alpha, \text{ where } L(y) = \frac{\gamma^2(y)(\varphi(y)-1)^2(\varphi(y)-\tau)^2}{(1-\varepsilon)(\varphi(y)-\tau)^2 + \varepsilon\tau^2(\varphi(y)-1)^2}, \varepsilon = \frac{1}{M}. \quad (11)$$

When $M \rightarrow \infty$, the quantity ε tends to zero. However, we cannot simply take $\varepsilon = 0$, since (at least for $\tau > 1$) for some value of y the denominator of $L(y)$ necessarily vanishes. Then it is impossible to cancel identical functions in the

numerator and denominator of $L(y)$. So, we analyze Eq. (11) for a small, but finite value of ε and then we tend it to zero. This way of analysis is correct.

When $\tau \leq 1$, the function $\varphi(y) - \tau$ nowhere vanishes, and we can simply take $\varepsilon = 0$. In this case the expression for $L(y)$ turns into $\gamma^2(\varphi - 1)^2$. Then Eq. (11) transforms in Eq. (7) that corresponds to the Hopfield model. Consequently, until the value of τ belongs to the interval $\tau \in (0, 1]$ the following is true:

$$y_c^{(2)}(\tau) \equiv y_c \approx 1.511, \alpha_c^{(2)}(\tau) \equiv \alpha_c \approx 0.138, m_c^{(2)}(\tau) \equiv m_c \approx 0.967. \quad (12)$$

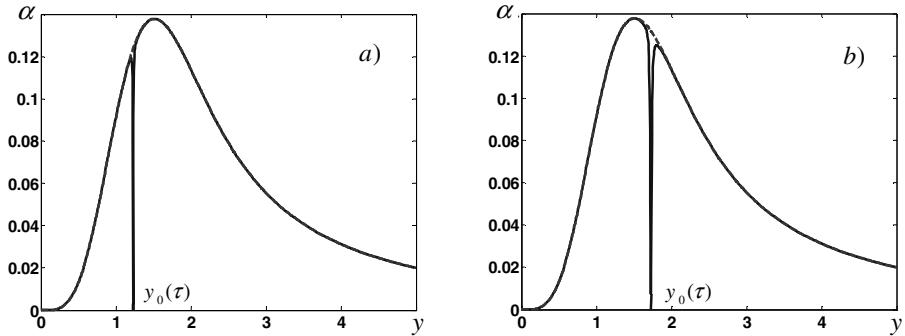


Fig. 2. The graph of the function $L(y)$ from Eq. (11), when $\varepsilon = 10^{-5}$ (solid line): a) when $\tau = 3$, the point $y_0(\tau)$ is on the left of $y_c \approx 1.511$; b) when $\tau = 10$, the point $y_0(\tau)$ is on the right of y_c . Dashed line shows the difference from the classical Hopfield model.

Let us examine the region $\tau > 1$. In this case the function $\varphi(y) - \tau$ vanishes at the point $y_0(\tau)$, whose value is determined by equation:

$$\varphi(y_0(\tau)) = \tau. \quad (13)$$

Out of the small vicinity of the point $y_0(\tau)$ the parameter ε in Eq. (11) can be tended to zero. At that the expression for the function $L(y)$ is as in the case of Hopfield model: $L(y) = \gamma^2(\varphi - 1)^2$. At the point $y_0(\tau)$ itself for small, but finite value of ε , the function $L(y)$ is equal to zero: $L(y_0(\tau)) = 0$. If y is from the small vicinity of the point $y_0(\tau)$ and it tends to $y_0(\tau)$, for any finite value of ε the curve $L(y)$ quickly drops to zero. Thus, for any finite value of ε the graph of the function $L(y)$ practically everywhere coincides with the curve $\gamma^2(\varphi - 1)^2$ that corresponds to the Hopfield model (7). And in the small vicinity of the point $y_0(\tau)$ the curve $L(y)$ has a narrow dip up to zero whose width is proportional to the value of ε .

As long as the weight $\tau < \varphi(y_c) \approx 5.568$, the point $y_0(\tau)$ is at the left of y_c . For this case in Fig. 2a we show the curve $L(y)$. The maximum of the curve $L(y)$

corresponds to the critical point $y_c \approx 1.511$ and it does not depend on τ . Consequently, the expressions (12) are justified not only for $\tau \in [0,1]$, but in the wider interval $0 < \tau \leq \varphi(y_c) \approx 5.568$.

On the contrary, for the values of the weight $\tau > 5.568$ the point $y_0(\tau)$ is on the right of y_c . For this case an example of the curve $L(y)$ is shown in Fig. 2b. The maximum point of the curve $L(y)$, which is interesting for us, coincides with the peak of the curve that is slightly on the right of the point $y_0(\tau)$. From continuity reasons it is evident that when $\varepsilon \rightarrow 0$ this peak shifts to the point $y_0(\tau)$. Consequently, when $\varepsilon \rightarrow 0$ we have for $\tau > 5.568$:

$$y_c^{(2)}(\tau) \equiv y_0(\tau), \quad m_c^{(2)}(\tau) \equiv \text{erf}(y_c^{(2)}(\tau)), \quad \alpha_c^{(2)}(\tau) = \frac{2}{\pi} (\tau - 1)^2 e^{-2y_0^2(\tau)}. \quad (14)$$

3.3 Computer Simulations

The obtained results were verified with the aid of computer simulations. For a given value of N the load parameter α was fixed. Then $M = \alpha N$ randomized patterns were generated, and they were used to construct the connection matrix with the aid of Eq. (2) and weights as in Eq. (8). When choosing the weight τ we proceeded from following reasons. Let α be a fixed value of the load parameter. Then we found the critical value of the weight $\tau(\alpha)$ for which the given α was a critical one (this can be done when solving Eqs. (9) and (10) simultaneously). We also defined the breakpoint $y_c(\alpha)$. If constructing the connection matrix with the weight τ that is equal to $\tau(\alpha)$, the mean value of the overlap of the local minimum with the pattern has to be close to $m_c(\alpha) = \text{erf}(y_c(\alpha))$. If the weight τ is less than $\tau(\alpha)$ the mean value of the overlap has to be close to 0. If the weight τ is larger than $\tau(\alpha)$, the mean value of the overlap has to be larger than $m_c(\alpha)$. Under further increase of τ the mean overlap has to tend to 1.

To verify the theory relating to the pattern with the individual weight $r_i = \tau$, three experiments have been done. In all experiments the dimensionality of the problem was $N = 10000$. For each load parameter α the mean value of overlap (with the single pattern) $\langle m \rangle$ was calculated with the aid of ensemble averaging. Ensemble consisted of 10 different matrices. For testing three values of α were chosen: $\alpha = 0.12$, $\tau(\alpha) \approx 0.944$, $m_c(\alpha) \approx 0.971$; $\alpha = 0.38$, $\tau(\alpha) \approx 1.501$, $m_c(\alpha) \approx 0.919$; $\alpha = 3.0$; for this value of α there is no jump of the overlap, but beginning from $\tau(\alpha) \approx 3.171$, the overlap has to increase smoothly. In Fig.3 the graphs for all three values of the load parameter are presented. Theoretical characteristics $m_c^{(1)}(\tau)$ are shown by dashed lines, and results of computer simulations are given by solid lines.

For $\alpha = 0.12$ (and $\alpha = 0.38$) on the experimental curves we clearly see expected jump of mean overlap that have place near critical value of the weight $\tau(\alpha) \approx 0.944$

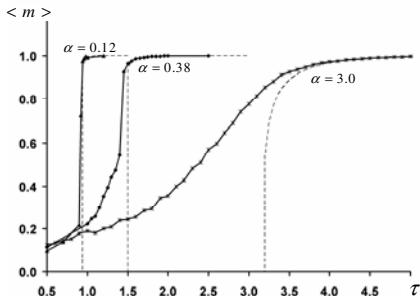


Fig. 3. Theory and experiments for the pattern with the individual weight $r_i = \tau$. The graphs correspond to three different values of the load parameter $\alpha = 0.12$, 0.38 and 3.0 . Solid lines are the results of experiments; dashed lines show theory.

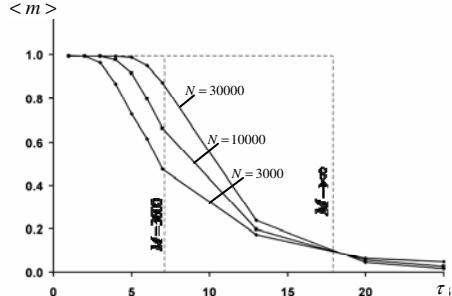


Fig. 4. Theory and experiments for patterns with the same weight $r_\mu \equiv 1$ when $\alpha = 0.12$. Solid lines show the results of experiments for three values of dimensionality N .

(and $\tau(\alpha) \approx 1.501$ respectively). At the left side the values of the mean overlap are not equal to zero. First, this can be due to not sufficiently large dimensionality of the problem. The point is that all theoretical results are related to the case $N \rightarrow \infty$. For our computer simulations we used large, but finite dimensionality N . Second, the theory is correct when the mean overlap is close to 1 (but not to 0). In this region of values of τ we have rather good agreement of the theory with computer simulation.

For the third load parameter $\alpha = 3.0$ we expected a smooth increasing of the mean overlap $\langle m \rangle$ from 0 to 1. Indeed, the last right solid curve in Fig. 3 increases smoothly without any jumps. However, according our theory the increasing ought to start beginning from $\tau \approx 3.1$, while the experimental curve differs from zero much earlier. This discrepancy between theory and experiment can be explained in the same way as it has been done in the end of the previous paragraph.

To verify our theory relating to patterns with equal weights $r_\mu \equiv 1$, $\mu \geq 2$ we used the same procedure. For the load parameter $\alpha = 0.12$ and several dimensionalities N ($N = 3000$, 10000 and 30000) we calculated the mean overlaps $\langle m \rangle$ of the local minima with the nearest patterns as function on τ . We averaged both over $M - 1$ patterns of the given matrix and over 10 randomized matrices constructed for the given value of τ .

According to the theory, for $\alpha = 0.12$ the breakdown of the overlap $\langle m \rangle$ has to take place when $\tau \approx 17.1$. In Fig. 4 this place is marked by the right vertical dashed straight line with the label “ $M \rightarrow \infty$ ”. If N and M really were infinitely large just in this place the breakdown of $\langle m \rangle$ had to take place. In Fig. 4 the real dependency of $\langle m \rangle$ on τ that was observed in our experiments was shown by three solid lines corresponding to different N .

Noticeable difference between the theory and computer simulations must not confuse us. Apparently this difference is due to finite dimensionalities of experimental connection matrices. Earlier computer verifications of classical theoretic

results faced just the same problems (see references in [4]). As a way out the authors usually extrapolated their experimental results into the region of very large N .

Note, when N increases the experimental curve in Fig. 4 tends to “theoretical step-function”, which is indicated with the aid of dashed line. A correction due to finite dimensionality of the problem can be taken into account if we insert the explicit expression $\varepsilon = 1/M$ in Eq. (11). Then for $N = 30000$ we have $M = \alpha N = 3600$. When this value of M is used in Eq. (11), we obtain that the breakdown of the overlap $\langle m \rangle$ has to take place not in the vicinity of $\tau \approx 17.1$, but much earlier when $\tau \approx 7.1$. The corresponding dashed line with the label “ $M = 3600$ ” is shown in Fig. 4. Its location noticeably better correlates with the start points of decrease of experimental curves.

3.4 Depths of Local Minima

Let us find out how the depth of local minimum depends on the value of τ . By the depth of the local minimum we imply the modulus of its energy (3): $|E|$. For different matrices the energies can be compared only if the same scale is used for their calculation. We normalized elements of each matrix dividing them by the square root from the dispersion of matrix elements: $\sigma_J = \sqrt{\tau^2 + \alpha N}$ is the standard deviation of matrix elements of quasi-Hebbian matrix J (2) for the weights (8). Then for a depth of a local minimum we obtain the expression

$$|E| = \frac{1}{\sigma_J} \sum_{\mu=1}^M r_\mu \left(m_\mu^2 - \frac{1}{N} \right).$$

Let us examine the local minimum in the vicinity of the pattern ξ^1 with the weight $r_1 = \tau$. For this minimum we obtain according to Eqs. (5)-(9)

$$|E_1| = \frac{1}{\sigma_J} (\tau m_1^2 + \sigma_1^2 - \alpha) , \quad \sigma_1 = \tau \gamma \varphi. \quad (15)$$

Here α is the load parameter, m_1 and σ_1 are calculated according to expressions (6), γ and φ are given by the expressions (5). Analysis of Eq. (15) and numerical calculations show that when τ increases the depth of the minimum increases too. While τ is less than a critical value $\tau_c(\alpha)$ the local minimum is very far from the pattern ($m_1 \approx 0$) and its depth is equal to zero. When $\tau > \tau_c(\alpha)$, minimum quickly approaches to the pattern ($m_1 \rightarrow 1$) and its depth increases in a jump-like way. After that $|E_1|$ increases as $\tau / \sqrt{\tau^2 + \alpha N}$. In Fig. 5 for different load parameters α the dependence of $|E_1|$ on τ is shown. When τ becomes very large ($\tau > \sqrt{\alpha N}$) the depth of the minimum asymptotically tends to 1. It could be thought that for such large τ the matrix J is constructed with the aid of only one pattern ξ^1 .

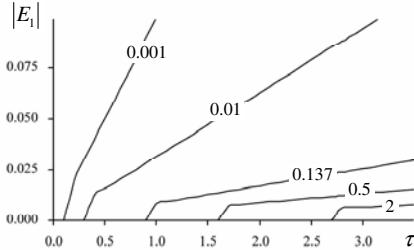


Fig. 5. The dependence of the depth of the local minimum from the vicinity of the pattern ξ^l (with the individual weight $r_i = \tau$) on τ . The load parameter α changes from 0.001 to 2.

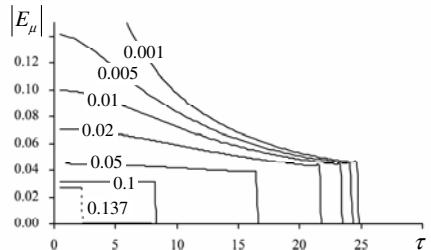


Fig. 6. The dependence of the depth of the local minimum from the vicinity of one of the patterns ξ^μ ($r_\mu = 1$) on τ . The load parameter α changes from 0.001 to 0.137.

Now let us examine local minima near patterns with the same weight $r_\mu = 1$ ($\mu \geq 2$). First, independent of the value of τ , these local minima exist only when $\alpha \leq \alpha_c$ (see Eqs. (12), (14)). The expression for the depth of the μ -th minimum has the form:

$$|E_\mu| = \frac{1}{\sigma_j} (m_\mu^2 + \sigma_\mu^2 - \alpha), \quad \sigma_\mu = \gamma \varphi. \quad (16)$$

Analysis of Eq. (16) shows that when τ increases, the depth of the local minimum decreases (see Fig.6). When τ becomes equal to some critical value $\tau_c(\alpha)$, the depth of the minimum reaches its minimal value $E_c = E_c(\alpha)$:

$$E_c = \frac{2}{\sqrt{\alpha_c N}} - \alpha_c \alpha.$$

For further increase of τ the overlap m_μ step-wise becomes equal to zero: minimum abruptly “goes away” from the pattern ξ^μ , and its depth $|E_\mu|$ drops to zero. From the point of view of the associative memory this means that for $\tau > \tau_c(\alpha)$ the memory of the network is destroyed.

4 Discussion and Conclusions

Local minima of the functional (1) are fixed points of an associative neural network. In this paper we examined minima localized near patterns, which were used to construct the connection matrix. All obtained results can be interpreted in terms of neural networks concepts, which are the storage capacity, patterns restoration and so on. From this point of view two results are of special interest.

First, notice that if $\tau \geq 3$ for the pattern with individual weight $r_i = \tau$ the phase transition of the first kind disappears. We recall that when τ increases the critical

value of the load parameter $\alpha_c^{(1)}$ also increases, the overlap of the local minimum with the pattern $m_c^{(1)}$ steadily decreases. Beginning from $\tau = 3$ the overlap in the critical point vanishes. This means that the phase transition ceases to exist. Second, it is interesting how characteristics of local minima located near patterns with weights $r_\mu = 1$ ($\mu \geq 2$) depend on the value of τ . We recall that at first increase in τ does not affect these local minima at all. While $\tau < 5.568$ neither the overlaps of the local minima with the patterns, nor its critical characteristics depend on τ (see Eq. (12)). It seems rather reasonable since the number of patterns with weights $r_\mu = 1$ is very large ($M \gg 1$). Therefore, as would be expected, the influence of only one pattern with the weight τ is negligible small. However, as soon as τ exceeds the critical value $\tau_c \approx 5.568$ (which is not so large) its influence on the huge number of local minima from the vicinities of patterns with weights $r_\mu = 1$ becomes rather noticeable (see Eqs. (13)). Let us emphasize that all the results are justified by computer simulations.

In conclusion we note that Hebbian connection matrix with different weights r_μ was examined previously. For example, in [5] were obtained the basic equation for this type of connection matrix. This equation coincides with our Eq. (4). However, then the authors of [5] simplified their equations and only the standard Hopfield model was analyzed.

References

1. Amit, D., Gutfreund, H., Sompolinsky, H.: Statistical Mechanics of Neural Networks Near Saturation. *Annals of Physics* 173, 30–67 (1987)
2. Hertz, J., Krogh, A., Palmer, R.: *Introduction to the Theory of Neural Computation*. Addison-Wesley, New York (1991)
3. Kryzhanovsky, B.V.: Expansion of a matrix in terms of external products of configuration vectors. *Optical Memory & Neural Networks (Information Optics)* 16(4), 187–199 (2007)
4. Frolov, A.A., Husek, D., Muraviev, I.P.: Informational efficiency of sparsely encoded Hopfield-like autoassociative memory. *Optical Memory & Neural Networks (Information Optic)* 12(3), 177–197 (2004)
5. van Hemmen, J.L., Kuhn, R.: Collective Phenomena in Neural Networks. In: Domany, E., van Hemmen, J.L., Shulten, K. (eds.) *Models of Neural Networks*, Springer, Berlin (1992)

A Learned Saliency Predictor for Dynamic Natural Scenes

Eleonora Vig¹, Michael Dorr^{1,2}, Thomas Martinetz¹, and Erhardt Barth¹

¹ Institute for Neuro- and Bioinformatics, University of Lübeck,
Ratzeburger Allee 160, 23538 Lübeck, Germany
{vig,dorr,martinetz,bARTH}@inb.uni-luebeck.de

² Schepens Eye Research Institute, Dept. of Ophthalmology, Harvard Medical School,
20 Staniford Street, Boston, MA 02114, USA
michael.dorr@schepens.harvard.edu

Abstract. We investigate the extent to which eye movements in natural dynamic scenes can be predicted with a simple model of bottom-up saliency, which *learns* on different visual representations to discriminate between salient and less salient movie regions. Our image representations, the geometrical invariants of the structure tensor, are computed on multiple scales of an anisotropic spatio-temporal multiresolution pyramid. Eye movement data is used to label video locations as salient. For each location, low-dimensional features are extracted on the multiscale representations and used to train a classifier. The quality of the predictor is tested on a large test set of eye movement data and compared with the performance of two state-of-the-art saliency models on this data set. The proposed model demonstrates significant improvement – mean ROC score of 0.665 – over the selected baseline models with ROC scores of 0.625 and 0.635.

Keywords: eye movements, visual saliency, low-level image properties, natural dynamic scenes, structure tensor.

1 Introduction

The active nature of seeing has been of great interest to both the biological and computer vision community. In human vision, the environment is sampled with 2-3 gaze shifts per second, by foveating relevant, so called “salient” items in the scene. Insights into the processes that guide eye movements towards relevant targets are of importance also for their utility in various image processing applications such as image and video compression [5][6] and active vision [1]. Such visual processes are believed to be influenced by two types of factors: on the one hand, eye movements are driven involuntarily, by stimulus “salience”; on the other hand, they are also influenced by the task at hand and contextual knowledge [17]. Here, we are dealing with the former, stimulus-driven, type of visual attention.

Over the recent years, a number of studies have investigated the relationship between eye movements and low-level image features at fixations¹, e.g., [12,13]. These so-called bottom-up models of attention center on the concept of a “saliency map”, which topographically encodes stimulus conspicuity [11]. Inspired by the Feature Integration Theory of Treisman and Gelade [14], in the first stage of visual processing separate low-level features such as edges, contrast, or color are extracted on multiple scales. Next, normalized center-surround difference maps are computed for individual features and combined together to obtain the global saliency map. From this map, the next location to be fixated is chosen by a selection process, which combines winner-take-all and inhibition-of-return strategies.

Most existing bottom-up saliency models are biologically inspired and they differ in their underlying computational principles – mainly from information theory – they use to formally define the concept of saliency: information maximization [18,3], Bayesian surprise [6], self-information of visual features [19], efficient coding [15], and optimal decision making under constraints of computational parsimony [4]. Since these biological models tend to be complex, a large number of parameters need to be tuned manually.

Learning techniques are often employed as a practical solution to the parameter tuning problem. Still, only very recently, approaches to derive saliency-based interest operators from human fixation data have been pursued. In [8], the authors learned optimal parameters for an attention model based on low-, mid- and high-level features calculated by existing saliency methods. Kienzle et al., on the other hand, employed machine learning algorithms to learn directly on the pixel intensities of static scenes [10] and Hollywood videos [9], with the goal to find the structural differences between fixated and non-fixated locations. Although the model structure was inferred from the data and not set manually, predictability was constrained by the reduced ability of learning algorithms to operate in high-dimensional spaces given a limited number of training samples. To account for the noise in both the eye tracking and the biological system, studies usually need to consider a spatio-temporal neighborhood around the fixation. Therefore, feature space dimensionality, in which the learning is conducted, is determined by the number of pixels of the neighborhood around the fixation, which for a reasonably sized neighborhood (e.g., 64 by 64 pixels, 2.5 deg) grows rapidly (more than 4000 dimensions). Therefore, the algorithms in [10,9] were limited to a single spatial scale.

Most work on gaze allocation has dealt with static stimuli, and only recently has the number of studies dealing with videos increased. Incorporating temporal information is not always straightforward, and in a learning context the task of eye movement prediction is further complicated by the increased number of dimensions.

¹ In the process of seeing, our eyes alternate between fixations, when they are aimed at a fixed point, and rapid reorienting movements called saccades.

1.1 Overview of the Proposed Approach

In this paper, we combine machine learning techniques with structure tensor-based multi-scale image features² to predict eye movements on natural dynamic scenes. Figure II gives a graphical overview of the approach. We use a large set of eye movements to label video patches as either attended or non-attended. Our goal is to learn the structural differences between these two classes of space-time patches, and apply the learned model to predict the saliency (or class membership) of novel, unlabeled video regions. To do so, we compute low-level feature maps of the videos, derived from the tensor invariants, on several spatio-temporal scales of an anisotropic image pyramid. Next, instead of learning on the high-dimensional feature patches (e.g., 64 by 64 pixels on a single scale), we use a simple method to reduce dimensionality. In the neighborhood of each fixation, we extract the local feature energy: the root-mean-square of the pixels in the spatio-temporal feature patch. We compute such an average energy (a single scalar) on each scale of the feature pyramids. Thus, for each fixation, we obtain a low-dimensional representation consisting of the energy values on the different scales. Together with the class labels (attended or non-attended), the feature energy vectors form the training data of a classifier, which learns a mapping between feature energy vectors and their class membership (i.e., salient or not). Finally, we use ROC analysis to test the model’s predicting power on unlabeled test data.

In [15], an earlier version of this model was used to predict gaze behavior of new observers on videos that have already been “seen” (i.e., learned on) by the classifier. In other words, the complete movie set was used both for training and testing, but eye movement data of any particular viewer were only present in either the training or the test set. In this paper, the model is put to the test in terms of how well it can predict eye movements on new, unseen videos. Furthermore, we extend this model by extracting the image features (geometrical invariants) on an anisotropic instead of an isotropic pyramid, which incorporates more information at the cost of moderate increase in the number of dimensions. Also, improvements are suggested on how the negative examples, i.e., locations that were not fixated, are labeled. Finally, the approach is placed in the context of existing methods: the model’s performance is compared to the predictive power of two state-of-the-art saliency models of dynamic scenes [719].

2 Methods

2.1 Geometrical Invariants

The original video representation is considered to be highly redundant. Therefore, learning on some low-level properties of a movie that are known to be predictive for saliency is beneficial, as it removes the redundancy which appears

² Features here denote some low-level quantifiable properties of the image/video, such as color, edginess, contrast, etc.

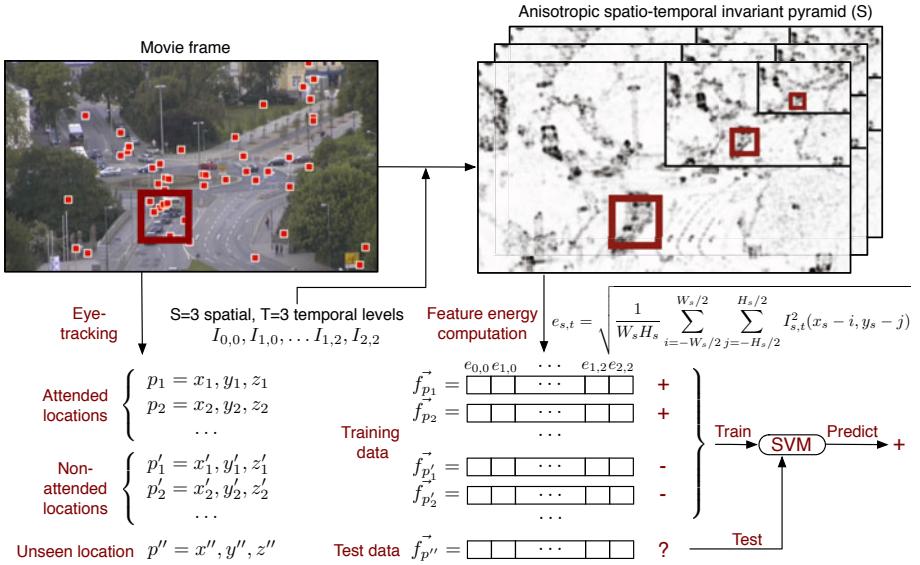


Fig. 1. Flow diagram summarizing our approach. Using eye tracking data (fixations are denoted by small filled squares in the movie frame from the left), we label movie regions as attended or non-attended. Image features – the geometrical invariants – are extracted on multiple scales of an anisotropic spatio-temporal pyramid. For a neighborhood (large unfilled square shown schematically) around each location, the average feature energy is computed on each scale of the spatio-temporal pyramid. An SVM is trained on the obtained energy vectors and is then used to predict whether an unseen location will be attended or not.

as noise to the classifier. Thus, instead of learning on the raw image intensities, we choose to use structure-tensor based image representations for our analysis. The structure tensor has been extensively used in image processing for solving a number of problems from motion and orientation estimation to occlusion detection, etc. Its geometrical invariants have been proven to be good bottom-up predictors of eye movements [15].

Considering the video as a function of space and time $f(x, y, t)$ ($f : \mathbb{R}^3 \rightarrow \mathbb{R}$), the structure tensor is defined as

$$\mathbf{J} = \int_{\Omega} \begin{bmatrix} f_x^2 & f_x f_y & f_x f_t \\ f_x f_y & f_y^2 & f_y f_t \\ f_x f_t & f_y f_t & f_t^2 \end{bmatrix} d\Omega , \quad (1)$$

where the integral over Ω is a Gaussian smoothing function and f_x , f_y , and f_t stand for the first order partial derivatives.

Typical image or movie structures can be characterized based on the geometrical invariants of the structure tensor, H , S , and K , which correspond to the minimum “intrinsic dimension” (iD – local signal variation) of a region:

$$\begin{aligned} H &= 1/3 \operatorname{trace}(\mathbf{J}) \\ S &= |M_{11}| + |M_{22}| + |M_{33}| \\ K &= |\mathbf{J}| \end{aligned} \quad (2)$$

where M_{ij} are minors of \mathbf{J} . If $K \neq 0$, the intrinsic dimension is 3, i.e., K encodes space-time corners and non-constant motion in the movie. If $S \neq 0$ the iD is at least 2, i.e., S additionally encodes stationary corners, and straight edges that change in time. Finally, if $H \neq 0$ the iD is at least 1, and H also responds to stationary edges, and uniform regions that change in time.

Previously, we have found that the degree to which eye movements can be predicted increases with the intrinsic dimension: the higher the intrinsic dimension, the higher the predictive power.

2.2 Multiscale Features

The above described invariants were computed on each scale of an anisotropic spatio-temporal multiresolution pyramid. As opposed to an isotropic pyramid, where space and time are subsampled simultaneously, in case of an anisotropic spatio-temporal pyramid each level of a spatial pyramid is decomposed further into its temporal bands. The resulting finer partition of the spectrum allows for the consideration of more information, but also comes at a computational cost. Partial derivatives of J in Eq. 11 were calculated by first smoothing the video with spatio-temporal 5-tap binomial kernels, and then applying $[-1, 0, 1]$ kernels. The smoothing kernel Ω was another 5-tap binomial. The details of the computation of invariants are the same as in [15] but for the fact that here an anisotropic pyramid is used.

2.3 Dimensionality Reduction

In the introduction, we argued that, on the one hand, models of saliency have to consider a spatio-temporal neighborhood (or “patch”) around the fixation to account for location uncertainty. On the other hand, in a machine learning context, even small neighborhood sizes quickly become intractable; therefore, in practice, learning can only be performed on a single scale. To overcome this problem and allow incorporating information from multiple scales, we “compress” raw pixel information by averaging image feature energy in the neighborhood around the fixation, so that we obtain a single scalar value for the whole neighborhood. This allows us now to compute such feature energy on every scale of the above spatio-temporal pyramids, as the dimensionality is still kept low. Here, we use a spatial neighborhood only, as the uncertainty induced by measurement errors and saccade imprecision is higher in the spatial domain than in the temporal one.

For an attended (or not attended) movie location $p = (x, y, z)$ (with spatial coordinates x and y , and frame number z), we compute a feature vector $\mathbf{f}_p = (e_{0,0}, e_{0,1}, \dots, e_{S-1,T-1})$ consisting of the feature energies extracted on each scale of an anisotropic pyramid with S spatial and T temporal levels. The feature

energy of a neighborhood (centered around the location p) computed on the s -th spatial and t -th temporal pyramid level is thus

$$e_{s,t} = \sqrt{\frac{1}{W_s H_s} \sum_{i=-W_s/2}^{W_s/2} \sum_{j=-H_s/2}^{H_s/2} I_{s,t}^2(x_s - i, y_s - j)}, \quad (3)$$

where W_s and H_s stand for the (subsampled) spatial width and height of the neighborhood on the s -th spatial scale (independent of the temporal scale). $I_{s,t}$ represents the s -th spatial and t -th temporal level of one of the invariant pyramids, H , S , and K . The spatial coordinates of the location are also subsampled on the spatial scale s : $(x_s, y_s) = (x/2^s, y/2^s)$.

2.4 Learning

Given a collection of videos together with a set of attended and not attended locations on these videos, we can now formulate the task of predicting salient locations as a binary decision problem, allowing to apply efficient classification algorithms from machine learning.

Thus, the problem of learning salient locations consists in finding, based on the locations' feature energy vectors and associated class labels (attended or not) $(\mathbf{f}_{p_i}, l_i) \in \mathbb{R}^{S \times T} \times \{-1, 1\}$, a function $g : \mathbb{R}^{S \times T} \rightarrow \mathbb{R}$, that is returning for a previously unseen movie patch \mathbf{p} , based on its energy vector \mathbf{f}_p , a confidence value quantifying the patch's level of saliency.

The data is partitioned “movie-wise” into a training and a test set: gaze data of all viewers on one movie are retained for testing, while the fixations on the remaining movies are used for the training. Thus, we are predicting gaze behavior on new movies that the classifier has not yet “seen”.

For the classification we use a standard soft margin Support Vector Machine with Gaussian kernels. Prior to training, we linearly scale each attribute to $[-1, 1]$. Optimal model parameters are found with cross-validation on the training sequence. To measure the quality of prediction, we perform an ROC analysis using the collected human gaze data as ground truth.

3 Experimental Evaluation

The following section evaluates the predictive power of the proposed approach as compared with two standard models of bottom-up saliency for videos: that of Itti and Koch [7], and SUNDAY [19]. The saliency model of Itti & Koch is perhaps the most well-known model, against which all other approaches are compared. It is an implementation of the classical saliency map described in the introduction. SUNDAY uses a Bayesian framework to analyze fixations: novelty is defined as self-information of the visual features, but the natural statistics used to detect outliers are learned from previous examples, and are not based only on the current movie.

3.1 Videos and Eye Tracking Data

For our analysis, eye movements were acquired from 54 human subjects free-viewing 18 high-resolution (HDTV standard, 1280×720 pixels, 29.97 Hz) videos of natural outdoor scenes of about 20 s duration each. The video clips depicted populated streets and roundabouts, people in a pedestrian area, on the beach, playing in a park, and animals. They were presented on a screen covering 48 by 27 degrees of visual angle, so that the maximum displayed frequency was 13.3 cycles per degree. The recordings were conducted in Karl Gegenfurtner's lab at the Dept. of Psychology of Giessen University using an SR Research EyeLink II eye tracker running at 250 Hz. About 40,000 saccades were extracted from the raw gaze data using a dual-threshold velocity-based procedure [2].

3.2 Data Set Labeling

Whereas the class of salient locations is well defined by the set of fixations (more precisely by the locations where saccades land), the selection of non-fixated locations is not trivial. A seemingly intuitive choice would be to generate random locations from a uniform distribution either from the entire sequence or from regions that were not fixated. However, this method ignores a common phenomenon inherent in such data sets: the central fixation bias [13]. A tendency of human subjects is observed to fixate more in the central part of the display rather than in the periphery. Therefore, it has been proposed that negative examples should also be drawn from this specific distribution of human fixation locations. To assure identical distribution, a standard method used in the vision literature [13] is to consider (temporally aligned) fixations of another, randomly selected video, so that the negative training examples of movie A are chosen using the scanpaths of a randomly selected movie B. Previously, we followed this approach to generate an equal number of (about 40,000) non-attended locations. However, a drawback of this approach is that the negative set of locations, i.e., the fixations on a single random movie are clustered within the spatio-temporal volume of the movie because of the spatio-temporal coherence of eye movements on natural scenes. Thus, comparing two clustered set of movie locations leads to a slight overestimation of the prediction performance. Here, we corrected for this effect, by considering randomly selected scanpaths from randomly chosen movies, rather than taking all scanpaths of only one random video. This procedure ensures a more scattered distribution of the non-attended locations.

3.3 Experimental Results

To compute the geometrical invariants on different spatio-temporal scales, we constructed an anisotropic pyramid with $S = 3$ spatial and $T = 3$ temporal levels, as described in Section 2.2. Thus, for each attended and non-attended location (about 40,000 per class) we obtained a 9-dimensional feature energy vector. Feature energy was computed in a neighborhood of 128 by 128 pixels on the highest scale and accordingly smaller sizes on lower scales, so that the

Table 1. ROC scores of the different models

Model	ROC score	std
Chance	0.5	—
Itti & Koch	0.625	0.003
SUNDAY	0.635	0.002
SVM with H	0.647	0.003
SVM with S	0.650	0.002
SVM with K	0.665	0.002

effective window size was about 4.8 deg on all scales. These two design parameters were found by systematically evaluating, in terms of ROC analysis, a range of different numbers of pyramid levels and window sizes.

A classifier was trained on all but one video from the movie set and testing was performed on the withheld movie. The optimal parameters of the kernel SVM (i.e., the width of the Gaussian γ and the penalty term C) were found by 8-fold cross-validation on the training sequence. Testing was repeated 18 times so that each movie served as test data once. Next, we created a single ROC curve for the complete set of movies, by varying a threshold on the decision values returned for the 18 test sets. This analysis was performed for all three invariants of the structure tensor.

For comparison, the saliency maps computed by the two state-of-the-art algorithms were treated as maximum likelihood binary classifiers for discriminating between fixated and non-fixated movie locations. To create such maps, the default model parameters were used, detailed on the web pages of the toolboxes³. For the Itti & Koch model, different fusion schemes of the individual feature maps into a master saliency map were tested. Best results were obtained with the Maxnorm normalization scheme, in which the fusion of the feature maps is based on normalized summation. By thresholding these maps, movie regions above the threshold were classified as salient. A systematic variation of the threshold, again on the complete movie set, resulted in a single ROC curve per model.

Based on the ROC curve, a single measure, called the ROC score (or the Area Under the Curve – AUC), provides an estimate of the prediction quality. ROC scores for the different models are shown in Table II. Performance is shown as mean \pm standard deviation over 5 realizations of the data set of negative locations (i.e., random pairing of scanpaths and movies). Note that the set of positive locations is well defined by the saccade end-points, i.e., has no random component.

A random classifier has an ROC score of 0.5, whereas an optimal predictor reaches an AUC of 1.0. As seen in Table II, all models have an average ROC score higher than chance. Predictability using the invariants reaches an ROC score of up to 0.665 (invariant K), which is significantly higher than the AUC values obtained for the Itti & Koch (0.625) and SUNDAY (0.635) models. Also, invariant

³ <http://ilab.usc.edu/toolkit/>
<http://mplab.ucsd.edu/~nick/NMPT/>

K outperforms the invariants H and S by a fair margin. This is, however, to be expected, as we have previously shown that prediction performance increases with the intrinsic dimension: movie regions that change in more spatio-temporal directions are more predictive than more uniform regions, because they are more informative, and therefore, draw attention.

4 Discussion and Conclusion

In this paper, we have presented a simple model of bottom-up visual saliency, which combines tensor-based multiscale image representations with machine learning to predict where people look in natural dynamic scenes. To enable the computation of visual features on multiple scales of an anisotropic multiresolution pyramid, we compressed the raw pixel information in the neighborhood around the fixation to a single value: the neighborhood's average feature energy. Certainly, the use of all available (i.e., uncompressed) information would favor the more accurate prediction, but the increased dimensionality poses a problem to existing machine learning algorithms: their performance drops when the data dimensionality is high relative to the number of training data. Thus, even though we are aware of the amount of information loss introduced by such an averaging, we argue that this significant loss is counterbalanced by the increased ability of classifiers to operate in low-dimensional spaces.

We obtained very favorable prediction results as compared with two standard methods. We argue that the efficiency of the proposed approach is due to the use of relevant image features computed on high-resolution videos, combined with state-of-the-art machine learning techniques which operate on a very large data set of eye movements. Note that the model of Itti & Koch, and SUNDAY do not profit from the advantages of a large data set. Furthermore, our method is robust with respect to varying video content.

We have, however, restricted ourselves to a single type of image feature, namely the geometrical invariants of the structure tensor. We expect that an extension of the model to incorporate a number of different features, such as color and orientation, would positively influence the predictability even further.

Finally, note that eye movements are more predictable than indicated by our ROC scores since here we have removed a number of biases, used only a very low-dimensional representation, and did not consider temporal correlations of the scanpaths.

Acknowledgments. We would like to thank Karl Gegenfurtner: data were collected in his lab at the Dept. of Psychology of Giessen University. Our research has received funding from the European Commission within the project GazeCom (contract no. IST-C-033816) of the 6th Framework Programme. All views expressed herein are those of the authors alone; the European Community is not liable for any use made of the information.

References

1. Aloimonos, Y., Weiss, I., Bandyopadhyay, A.: Active Vision. *International Journal of Computer Vision* 1(4), 333–356 (1988)
2. Böhme, M., Dorr, M., Krause, C., Martinetz, T., Barth, E.: Eye Movement Predictions on Natural Videos. *Neurocomputing* 69(16–18), 1996–2004 (2006)
3. Bruce, N., Tsotsos, J.K.: Saliency, Attention, and Visual Search: An Information Theoretic Approach. *Journal of Vision* 9(3), 1–24 (2009)
4. Gao, D., Vasconcelos, N.: Decision-Theoretic Saliency: Computational Principles, Biological Plausibility, and Implications for Neurophysiology and Psychophysics. *Neural Computation* 21(1), 239–271 (2009)
5. Geisler, W.S., Perry, J.S.: A Real-Time Foveated Multiresolution System for Low-bandwidth Video Communication. In: *Human Vision and Electronic Imaging: SPIE Proceedings*, pp. 294–305 (1998)
6. Itti, L., Baldi, P.: Bayesian Surprise Attracts Human Attention. *Vision Research* 49(10), 1295–1306 (2009)
7. Itti, L., Koch, C., Niebur, E.: A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(11), 1254–1259 (1998)
8. Judd, T., Ehinger, K., Durand, F., Torralba, A.: Learning to Predict Where Humans Look. In: *Proceedings of IEEE International Conference on Computer Vision* (2009)
9. Kienzle, W., Schölkopf, B., Wichmann, F.A., Franz, M.O.: How to Find Interesting Locations in Video: a Spatiotemporal Interest Point Detector Learned from Human Eye Movements. In: *Proceedings of the 29th Annual Symposium of the German Association for Pattern Recognition*, pp. 405–414. Springer, Berlin (2007)
10. Kienzle, W., Wichmann, F.A., Schölkopf, B., Franz, M.O.: A Nonparametric Approach to Bottom-Up Visual Saliency. In: *Advances in Neural Information Processing Systems*, pp. 689–696. MIT Press, Cambridge (2006)
11. Koch, C., Ullman, S.: Shifts in Selective Visual Attention: towards the Underlying Neural Circuitry. *Human Neurobiology* 4(4), 219–227 (1985)
12. Reinagel, P., Zador, A.M.: Natural Scene Statistics at the Centre of Gaze. *Network* 10, 341–350 (1999)
13. Tatler, B.W., Baddeley, R.J., Gilchrist, I.D.: Visual Correlates of Fixation Selection: Effects of Scale and Time. *Vision Research* 45, 643–659 (2005)
14. Treisman, A.M., Gelade, G.: A Feature-Integration Theory of Attention. *Cognitive Psychology* 12(1), 97–136 (1980)
15. Vig, E., Dorr, M., Barth, E.: Efficient Visual Coding and the Predictability of Eye Movements on Natural Movies. *Spatial Vision* 22(5), 397–408 (2009)
16. Wang, Z., Lu, L., Bovik, A.C.: Foveation Scalable Video Coding with Automatic Fixation Selection. *IEEE Transactions on Image Processing* 12(2), 243–254 (2003)
17. Yarbus, A.L.: *Eye Movements and Vision*. Plenum Press, New York (1967)
18. Zetzsche, C., Schill, K., Deubel, H., Krieger, G., Umkehrer, E., Beinlich, S.: Investigation of a Sensorimotor System for Saccadic Scene Analysis: an Integrated Approach. In: Pfeifer, R., Blumenberg, B., Meyer, J., Wilson, S. (eds.) *Proc. 5th Intl. Conf. Soc. Adaptive Behavior*, vol. 5, pp. 120–126. MIT Press, Cambridge (1998)
19. Zhang, L., Tong, M.H., Cottrell, G.W.: SUNDAY: Saliency Using Natural Statistics for Dynamic Analysis of Scenes. In: *Proceedings of the 31st Annual Cognitive Science Conference*, Amsterdam, Netherlands (2009)

Learning a Combination of Heterogeneous Dissimilarities from Incomplete Knowledge

Manuel Martín-Merino

Universidad Pontificia de Salamanca
C/Compañía 5, 37002, Salamanca, Spain
mmartinmac@upsa.es

Abstract. The performance of many pattern recognition algorithms depends strongly on the dissimilarity considered to evaluate the sample proximities. The choice of a good dissimilarity is a difficult task because each one reflects different features of the data. Therefore, different dissimilarities and data sources should be integrated in order to reflect more accurately which is similar for the user and the problem at hand.

In many applications, the user feedback or the a priori knowledge about the problem provide pairs of similar and dissimilar examples. This side-information may be used to learn a distance metric that reflects more accurately the sample proximities. In this paper, we address the problem of learning a linear combination of dissimilarities using side information in the form of equivalence constraints. The minimization of the error function is based on a quadratic optimization algorithm. A smoothing term is included that penalizes the complexity of the family of distances and avoids overfitting.

The experimental results suggest that the method proposed outperforms a standard metric learning algorithm and improves classification and clustering results based on a single dissimilarity.

1 Introduction

Pattern Recognition algorithms depend critically on the choice of a good dissimilarity [18]. A large variety of dissimilarities have been proposed in the literature [1]. However, in real applications no dissimilarity outperforms the others because each dissimilarity reflects often different features of the data [11]. So, instead of using a single dissimilarity it has been recommended in [9][2] to consider a linear combination of heterogeneous dissimilarities and data sources.

Several authors have proposed techniques to learn a linear combination of kernels (similarities) from the data [2][9][14][19]. These methods are designed for classification tasks and assume that the class labels are available for the training set. However, for certain applications such as Bioinformatics, domain experts provide only incomplete knowledge in the form of which pairs of samples or genes are related [6]. This a priori information can be incorporated via semi-supervised metric learning algorithms using equivalence constraints [3]. Thus, [18] proposed a distance metric learning algorithm that incorporates such

similarity/dissimilarity information using a convex programming approach. The experimental results show a significant improvement in clustering results. However, the algorithm is based on an iterative procedure that is computationally intensive particularly, for high dimensional applications. To avoid this problem, [3][7][16] presented more efficient algorithms to learn a Mahalanobis metric. However, these algorithms are not able to incorporate heterogeneous dissimilarities and rely on the use of the Mahalanobis distance that may not be appropriate for certain kind of applications [11][12].

Our approach considers that the integration of dissimilarities that reflect different features of the data should help to improve the performance of clustering and classification algorithms. To this aim, a linear combination of heterogeneous dissimilarities is learnt considering the relation between kernels and distances [13]. A learning algorithm is proposed to estimate the optimal weights considering the similarity/dissimilarity constraints available. The method proposed is based on a convex quadratic optimization algorithm and incorporates a smoothing term that penalizes de complexity of the family of distances avoiding overfitting.

The metric learning algorithm proposed has been applied to a wide range of practical problems. The empirical results suggest that the method proposed helps to improve pattern recognition algorithms based on a single dissimilarity and a widely used metric learning algorithm proposed in the literature.

This paper is organized as follows: Section 2 introduces the idealized metric considered in this paper, section 3 presents the algorithm proposed to learn a combination of dissimilarities from equivalence constraints. Section 4 illustrates the performance of the algorithm using several benchmark and real datasets. Finally, Section 5 gets conclusions and outlines future research trends.

2 Idealized Dissimilarity

Let $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ be the input patterns. We are given side-information in the form of pairs that are considered similar or dissimilar for the application at hand. Let \mathcal{S} and \mathcal{D} be the subset of pairs of patterns known to be similar/dissimilar defined as:

$$\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \text{ is similar to } \mathbf{x}_j\} \quad (1)$$

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \text{ is dissimilar to } \mathbf{x}_j\} \quad (2)$$

Let $\{d_{ij}^l\}_{l=1}^M$ be the set of heterogeneous dissimilarities considered. Each dissimilarity can be embedded in a feature space via the empirical kernel map introduced in [13]. Let K_{ij}^l be the kernel matrix that represents the dissimilarity matrix $(d_{ij}^l)_{i,j=1}^n$. The kernel function can be written as an inner product in feature space [15] $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ and therefore, it can be considered a similarity measure [16].

The ideal similarity (kernel) should be defined such that it becomes large for similar patterns and small for dissimilar ones. Mathematically, the ideal kernel is defined as follows:

$$k_{ij}^* = \begin{cases} \max_l \{k_{ij}^l\} & \text{If } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ \min_l \{k_{ij}^l\} & \text{If } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \end{cases} \quad (3)$$

The idealized kernel introduced in this paper is related to the one proposed by [2] for classification purposes: $k(x_i, x_j) = 1$ if $y_i = y_j$ and 0 otherwise, where y_i denotes the label of x_i . However, there are two differences that are worth to mention. First, the ideal kernel proposed by [2] doesn't take into account the structure and distribution of the data, missing relevant information particularly for clustering applications. The idealized kernel defined here, collect information from a set of similarity measures and hence considers the spatial distribution of the data. Second, the kernel proposed by [2] can be considered an extreme case of the idealized kernel defined earlier and thus, more prone to overfitting.

Considering the relation between distances and kernels [15], the idealized distance between \mathbf{x}_i and \mathbf{x}_j can be written in terms of kernel evaluations as:

$$d^{2*}(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \quad (4)$$

$$= k^*(\mathbf{x}_i, \mathbf{x}_i) + k^*(\mathbf{x}_j, \mathbf{x}_j) - 2k^*(\mathbf{x}_i, \mathbf{x}_j) \quad (5)$$

The idealized dissimilarity improves the separability among different groups in the dataset. However, this dissimilarity may increase the overfitting. Hence, robust learning algorithms should be developed that achieve a balance between the complexity of the family of distances and the separability among different clusters.

3 Learning a Combination of Dissimilarities from Equivalence Constraints

In this section, we present a learning algorithm to estimate the optimal weights of a linear combination of kernels from a set of similarity/ dissimilarity constraints.

Let $\{k_{ij}^l\}_{l=1}^M$ be the set of kernels obtained from a set of heterogeneous dissimilarities via the empirical kernel map introduced in [13]. If non-linear kernels with different parameter values are considered, we get a wider family of measures that includes non-linear transformations of the original dissimilarities. The kernel sought is defined as:

$$k_{ij} = \sum_{l=1}^M \beta_l k_{ij}^l, \quad (6)$$

where the β_l coefficients are constrained to be ≥ 0 . This non-negative constraint on the weights helps to interpret the results and guarantees that provided all the individual kernels are positive semi-definite the combination of kernels is convex and positive semi-definite [14].

The optimization problem in the primal may be formulated as follows:

$$\min_{\beta, \xi} \frac{1}{2} \|\beta\|^2 + \frac{C_S}{N_S} \sum_{(x_i, x_j) \in S} \xi_{ij} + \frac{C_D}{N_D} \sum_{(x_i, x_j) \in D} \xi_{ij} \quad (7)$$

$$\text{s. t. } \beta^T \mathbf{K}_{ij} \geq K_{ij}^* - \xi_{ij} \quad \forall (x_i, x_j) \in S \quad (8)$$

$$\beta^T \mathbf{K}_{ij} \leq K_{ij}^* + \xi_{ij} \quad \forall (x_i, x_j) \in D \quad (9)$$

$$\beta_l \geq 0 \quad \xi_{ij} \geq 0 \quad \forall l = 1, \dots, M \quad (10)$$

where the first term in equation (7) is a regularization term that penalizes the complexity of the family of distances, C_S and C_D are regularization parameters that give more relevance to the similarity or dissimilarity constraints. N_S , N_D are the number of pairs in \mathcal{S} and \mathcal{D} , $\mathbf{K}_{ij} = [K_{ij}^1, \dots, K_{ij}^M]^T$, K_{ij}^* is the idealized kernel matrix introduced in section 2 and ξ_{ij} are the slack variables that allows for errors in the constraints.

To solve this constrained optimization problem the method of Lagrange Multipliers is used. The dual problem becomes:

$$\max_{\alpha_{ij}, \gamma} -\frac{1}{2} \sum_{\substack{(x_i, x_j) \in S \\ (x_k, x_l) \in S}} \alpha_{ij} \alpha_{kl} \mathbf{K}_{ij}^T \mathbf{K}_{kl} - \frac{1}{2} \sum_{\substack{(x_i, x_j) \in D \\ (x_k, x_l) \in D}} \alpha_{ij} \alpha_{kl} \mathbf{K}_{ij}^T \mathbf{K}_{kl} \quad (11)$$

$$+ \sum_{\substack{(x_i, x_j) \in S, \\ (x_k, x_l) \in D}} \alpha_{ij} \alpha_{kl} \mathbf{K}_{ij}^T \mathbf{K}_{kl} - \sum_{(x_i, x_j) \in S} \alpha_{ij} \gamma^T \mathbf{K}_{ij} - \frac{1}{2} \gamma^T \gamma \quad (12)$$

$$+ \sum_{(x_i, x_j) \in D} \alpha_{ij} \gamma^T \mathbf{K}_{ij} + \sum_{(x_i, x_j) \in S} \alpha_{ij} K_{ij}^* - \sum_{(x_i, x_j) \in D} \alpha_{ij} K_{ij}^*, \quad (13)$$

subject to:

$$0 \leq \alpha_{ij} \leq \begin{cases} \frac{C_S}{N_S} & \text{for } (x_i, x_j) \in \mathcal{S} \\ \frac{C_D}{N_D} & \text{for } (x_i, x_j) \in \mathcal{D} \end{cases} \quad (14)$$

$$\gamma_l \geq 0 \quad \forall l = 1, \dots, M, \quad (15)$$

where α_{ij} and γ_l are the lagrange multipliers. This is a standard quadratic optimization problem similar to the one solved by the SVM. The computational burden does not depend on the dimensionality of the space and it avoids the problem of local minima.

Once the α_{ij} and γ_l are computed, the weights β_l can be obtained considering $\partial L / \partial \beta = 0$:

$$\beta = \sum_{(x_i, x_j) \in S} \alpha_{ij} \mathbf{K}_{ij} - \sum_{(x_i, x_j) \in D} \alpha_{ij} \mathbf{K}_{ij} + \gamma. \quad (16)$$

The weights β_l can be substituted in equation (6) to get the optimal combination of heterogeneous kernels. Next, a clustering or classification algorithm based on kernels may be applied.

The metric learning algorithm proposed allows to incorporate into kernel clustering algorithms a priory knowledge given in the form of equivalence constraints.

However, it can also be applied to integrate a linear combination of heterogeneous dissimilarities and data sources into kernel based classifiers.

Several techniques are related to the one proposed here. In [18] it has been proposed an algorithm to learn a full or diagonal Mahalanobis metric from similarity information. The optimization algorithm is based on an iterative procedure that is more costly particularly for high dimensional problems. [3] and [7][16] have proposed more efficient algorithms to learn a Mahalanobis metric from equivalence constraints. The first one (Relevant Component Analysis), can only take into account similarity constraints and when there is no labels, the chunklets created from equivalence constraints are not well defined. All of them, rely solely on a Mahalanobis metric that may fail to reflect appropriately the sample proximities for certain kind of applications [11][12]. Hence, they are not able to integrate heterogeneous measures that convey complementary information. Finally, [17] has proposed a modification of the maximum margin clustering that is able to learn a linear combination of kernels. However, this algorithm is unsupervised and can not incorporate a priory information in a semi-supervised way. Besides, it can not be extended to other clustering and classification algorithms based on dissimilarities or kernels as the method proposed here.

4 Experimental Results

The algorithm proposed has been evaluated considering a wide range of applications. First, several clustering problems have been addressed. Table 1 shows the features of the different datasets. We have chosen problems with a broad range of signal to noise ratio (Var/Samp.), varying number of samples and classes. The first three problems correspond to benchmark datasets obtained from the UCI database <http://archive.ics.uci.edu/ml/datasets/>. The last ones aim to the identification of complex human cancer samples using the gene expression profiles. They are available from bioinformatics2.pitt.edu.

Finally, we apply the metric learning algorithm proposed to the prediction of protein subcellular location [10] considering a set of heterogeneous data sources. We use as a gold standard the annotation provided by the MIPS comprehensive yeast genome database (CYGD). CYGD assigns subcellular locations to 2138 yeast proteins. The primary input for the learning algorithm is a collection of seven kernel matrices obtained from different data sources. For a detailed description of the sources and kernels employed see [10].

Table 1. Features of the different datasets considered

	Samples	Variables	Var./Samp.	Classes
Wine (UCI)	177	13	0.17	3
Ionosphere (UCI)	351	35	0.01	2
Breast Cancer (UCI)	569	32	0.056	2
Lymphoma	96	4026	41.9	2
Colon Cancer	62	2000	32	2

All the datasets have been standardised subtracting the median and dividing by the inter-quantile range.

For high dimensional problems such as gene expression datasets, dimension reduction helps to improve significantly the clustering results [8]. Therefore, for the algorithms based on a single dissimilarity we have considered different number of genes 280, 146, 101, 56 and 34 obtained by feature selection [12]. We have chosen the subset of genes that gives rise to the smallest error. Genes have been ranked according to the interquartile range. As we are addressing clustering problems, feature selection methods that take into account the class labels are discarded. Regarding the algorithm proposed to integrate several dissimilarities, we have considered all the dissimilarities obtained for the whole set of dimensions.

The similarity/dissimilarity constraints are obtained as in [18]. \mathcal{S} is generated by picking a random subset of all pairs of points sharing the same class label. The size is chosen such that the number of connected components is roughly 20% of the size of the original dataset. \mathcal{D} is chosen in a similar way although the size in this case is less relevant. Twenty independent random samples for \mathcal{S} and \mathcal{D} are considered and the average error is reported.

Regarding the value of the parameters, the number of clusters is set up to the number of classes, C_S and C_D are regularization parameters and the optimal value is determined by cross-validation over the subset of labeled patterns. Finally, kernel k -means is restarted randomly 20 times and the errors provided are averages over 20 independent trials.

Clustering results have been evaluated considering two objective measures. The first one is the accuracy. It determines the probability that the clustering agrees with the “true” clustering in the sense that the pair of patterns belong to the same or different clusters. It has been defined as in [18]:

$$\text{accuracy} = \sum_{i>j} \frac{1\{1\{c_i = c_j\} = 1\{\hat{c}_i = \hat{c}_j\}\}}{0.5m(m-1)}, \quad (17)$$

where c_i is the true cluster label for pattern x_i , \hat{c}_i is the corresponding label returned by the clustering algorithm and m is the number of patterns. One problem of the accuracy is that the expected value for two random partitions is not zero. Therefore, we have computed also the adjusted randindex defined in [4] that avoids this problem. This index is also normalized between zero and one and larger values suggest better clustering.

Tables 2 and 3 show the accuracy and the adjusted randindex for the clustering algorithms evaluated. We have compared with a standard metric learning strategy proposed by [18], k -means clustering based on the Euclidean distance and k -means considering the best dissimilarity out of ten different measures. Both tables indicates which is the best distance for each case.

From the analysis of tables 2 and 3, the following conclusions can be drawn:

- The combination of dissimilarities improves significantly a standard metric learning algorithm for all the datasets considered. Our method is robust to overfitting and outperforms the algorithm proposed by Xing [18] in high

Table 2. Accuracy for k -means clustering considering different dissimilarities. The results are averaged over twenty independent random subsets \mathcal{S} and \mathcal{D} .

Technique		Kernel	Wine	Ionosphere	Breast	Colon	Lymphoma
k -means (Euclidean)	linear	0.92	0.72	0.88	0.87	0.90	
	pol. 3	0.87	0.73	0.88	0.88	0.90	
k -means (Best diss.)	linear	0.94	0.88	0.90	0.88	0.94	
	pol. 3	0.94	0.88	0.90	0.88	0.93	
		χ^2	Maha.	Manha.	Corr./euclid.		χ^2
Comb. dissimilarities	linear	0.94	0.90	0.92	0.89	0.95	
	pol. 3	0.96	0.89	0.92	0.90	0.92	
Metric learning (Xing)	linear	0.87	0.74	0.85	0.87	0.90	
	pol. 3	0.51	0.74	0.86	0.88	0.90	

Table 3. Adjusted RandIndex for k -means clustering considering different dissimilarities. The results are averaged over twenty independent random subsets \mathcal{S} and \mathcal{D} .

Technique		Kernel	Wine	Ionosphere	Breast	Colon	Lymphoma
k -means (Euclidean)	linear	0.79	0.20	0.59	0.59	0.65	
	pol. 3	0.67	0.21	0.60	0.59	0.65	
k -means (Best diss.)	linear	0.82	0.58	0.66	0.59	0.77	
	pol. 3	0.81	0.58	0.66	0.59	0.76	
		χ^2	Maha.	Manha.	Corr./euclid.		χ^2
Comb. dissimilarities	linear	0.82	0.63	0.69	0.60	0.79	
	pol. 3	0.85	0.60	0.69	0.63	0.73	
Metric learning (Xing)	linear	0.68	0.23	0.50	0.54	0.66	
	pol. 3	0.50	0.23	0.52	0.58	0.65	

dimensional datasets such as Colon cancer and Lymphoma. These datasets exhibit a high level of noise. We can explain this because the algorithm based on a combination of dissimilarities allows to integrate distances computed for several dimensions discarding the noise and reducing the errors.

- The combination of dissimilarities improves usually kernel k -means based solely on the best dissimilarity. This suggests that the integration of several dissimilarities allows to extract complementary information that may help to improve the performance. Besides, the algorithm proposed always achieves at least the same performance that k -means based on the best dissimilarity. Only for lymphoma and polynomial kernel we get worst results, probably because the value assigned to the regularization parameters overfit the data. We remark that the algorithm proposed, helps to overcome the problem of choosing the best dissimilarity, the kernel and the optimal dimension. This a quite complex and time consuming task for certain applications such as Bioinformatics.

Finally, the combination of dissimilarities improves always the standard k -means clustering based on the Euclidean measure.

- Tables 2 and 3 show that the best distance depends on the dataset considered. Moreover, we report that the performance of k -means depends strongly on the particular measure employed to evaluate the sample proximities.

Figure 1 shows a boxplot diagram for the accuracy and adjusted randindex coefficients. Twenty independent random samples for \mathcal{S} and \mathcal{D} are considered. Odds numbers correspond to the combination of dissimilarities and the even ones to the metric learning algorithm proposed by Xing. We can see that the differences between the method proposed here and the one proposed by Xing are statistically significant at 95% confidence level for all the datasets considered.

Regarding the identification of membrane protein classes, a linear combination of seven heterogeneous kernels (data sources) is learnt considering only similarity constraints. The size of \mathcal{S} is chosen such that the number of connected components is roughly 10% of the number of patterns. Once the kernel is learnt, a k -NN algorithm is run and the accuracy is estimated by ten-fold cross-validation. We have compared with k -NN based solely on a single data source and with the Lanckriet formalism [9]. Notice that the method proposed by Lanckriet is not able to work from similarity constraints only and needs the class labels. Besides, it is only applicable for SVM classifiers.

Table 4 shows that our algorithm improves k -NN based on a single kernel (source) by at least 4%. The β_l coefficients are larger for the diffusion and Hydrophobicity FFT kernels which is consistent with the analysis of [9] that

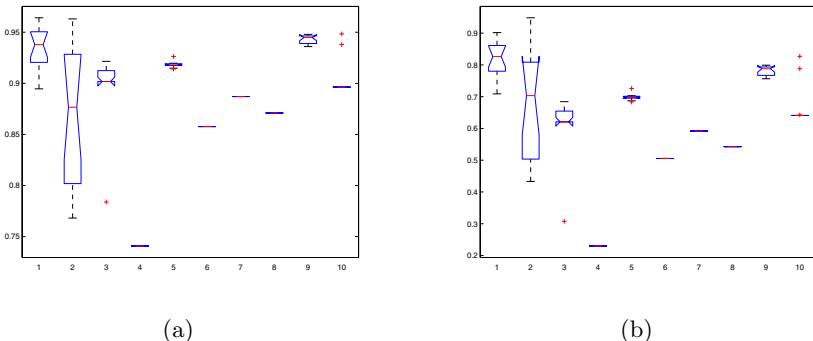


Fig. 1. Boxplots that compare the combination of dissimilarities with the metric learning algorithm proposed by Xing according to (a) accuracy and (b) Adjusted RandIndex. All the boxplots consider linear kernels.

Table 4. Accuracy of k -NN considering the best data sources and learning the optimal weights of a combination of heterogeneous data sources. Only five sources with non-null β_l are shown.

Source	Gen. Expre.	BLAST	Pfam	Hydropho.	Difussion	Combination	Lanckriet
Accuracy	73.30%	79.98%	82.48%	77.01%	80.16%	86.68%	88.66%
β_l	0.24	0.15	0.29	0.62	4.45	-	-

suggests that diffusion kernels perform the best. Kernels removed correspond to redundant kernels and not to the less informative as suggested by [5]. Our method performs similarly to the algorithm proposed by Lanckriet although we use only 10% of similarity constraints. Finally, we have incorporated a random kernel (source) to check the robustness against noise. The accuracy is similar (86.75%) which suggests that the combination algorithm tolerates high level of noise.

5 Conclusions

In this paper, we propose a semi-supervised algorithm to learn a combination of dissimilarities from equivalence constraints. The error function includes a penalty term that controls the complexity of the family of distances considered and the optimization is based on a robust quadratic programming approach that does not suffer from the problem of local minima.

The experimental results suggest that the combination of dissimilarities improves almost always the performance of clustering and classification algorithms based solely on a single dissimilarity. Besides, the algorithm proposed improves significantly a standard metric learning algorithm for all the datasets considered in this paper and is robust to overfitting.

Future research trends will focus on the application of this formalism to other problems such as gene function prediction.

References

1. Cox, T.F., Cox, M.A.A.: Multidimensional scaling, 2nd edn. Chapman & Hall/CRC (2001)
2. Cristianini, N., Kandola, J., Elisseeff, J., Shawe-Taylor, A.: On the kernel target alignment. *Journal of Machine Learning Research* 1, 1–31 (2002)
3. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a Mahalanobis Metric from Equivalence Constraints. *Journal of Machine Learning Research* 6, 937–965 (2005)
4. Hubert, L., Arabie, P.: Comparing Partitions. *Journal of Classification*, 193–218 (1985)
5. Hulsman, M., Reinders, M.J.T., de Ridder, D.: Evolutionary Optimization of Kernel Weights Improves Protein Complex Comembership Prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 6(3), 427–437 (2009)
6. Huang, D., Pan, W.: Incorporating Biological Knowledge into Distance-Based Clustering Analysis of Microarray Gene Expression Data. *Bioinformatics* 22(10), 1259–1268 (2006)
7. Kwok, J.T., Tsang, I.W.: Learning with Idealized Kernels. In: Proceedings of the Twentieth International Conference on Machine Learning, Washington DC, pp. 400–407 (2003)
8. Jeffery, I.B., Higgins, D.G., Culhane, A.C.: Comparison and Evaluation Methods for Generating Differentially Expressed Gene List from Microarray Data. *BMC Bioinformatics* 7(359), 1–16 (2006)

9. Lanckriet, G., Cristianini, N., Barlett, P., El Ghaoui, L., Jordan, M.: Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research* 3, 27–72 (2004)
10. Lanckriet, G.R.G., De Bie, T., Cristianini, N., Jordan, M.I., Stafford Noble, W.: A statistical framework for genomic data fusion. *Bioinformatics* 20(16), 2626–2635 (2004)
11. Martín-Merino, M., Blanco, A.: A local semi-supervised Sammon algorithm for textual data visualization. *Journal of Intelligence Information System* 33(1), 23–40 (2009)
12. Martín-Merino, M., Blanco, A., De Las Rivas, J.: Combining Dissimilarities in a Hyper Reproducing Kernel Hilbert Space for Complex Human Cancer Prediction. *Journal of Biomedicine and Biotechnology*, 1–9 (2009)
13. Pekalska, E., Paclík, P., Duin, R.: A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research* 2, 175–211 (2001)
14. Soon Ong, C., Smola, A., Williamson, R.: Learning the kernel with hyperkernels. *Journal of Machine Learning Research* 6, 1043–1071 (2005)
15. Vapnik, V.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
16. Wu, G., Chang, E.Y., Panda, N.: Formulating distance functions via the kernel trick. In: ACM SIGKDD, Chicago, pp. 703–709 (2005)
17. Zhao, B., Kwok, J.T., Zhang, C.: Multiple Kernel Clustering. In: Proceedings of the Ninth SIAM International Conference on Data Mining, Nevada, pp. 638–649 (2009)
18. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance Metric Learning, with Application to Clustering with Side-Information. In: Advances in Neural Information Processing Systems, pp. 505–512. MIT Press, Cambridge (2003)
19. Xiong, H., Chen, X.-W.: Kernel-Based Distance Metric Learning for Microarray Data Classification. *BMC Bioinformatics* 7(299), 1–11 (2006)

A Bilinear Model for Consistent Topographic Representations

Urs Bergmann and Christoph von der Malsburg

Frankfurt Institute for Advanced Studies, 60438 Frankfurt a.M., Germany

Abstract. Visual recognition faces the difficult problem of recognizing objects despite the multitude of their appearances. Ample neuroscientific evidence shows that the cortex uses a topographic code to represent visual stimuli. We therefore develop a bilinear probabilistic model that learns transformations to build an invariant topographic code in an unsupervised way. Simulations for the simple over-complete linear case yield V1 like Gabor receptive fields that are arranged in orientation, frequency and position maps. For the computationally more powerful case of two control units, we show that an application to natural inputs placed at different positions and with a consistent relative transformation (e.g. rotation), leads to an invariant topographic output representation and hence a relative implementation of the transformation in the control units, i.e. the Gabor receptive fields are transformed accordingly.

1 Introduction

The sensory parts of the neocortex are in general topographically organized, e.g. retinotopy in the primary visual cortex, or tonotopy in the primary auditory cortex. For retinotopy, there is by now a long tradition in modeling the organization of these mappings, see e.g. [1]. While most of these studies focus on the phenomenological aspects of topography, orientation maps, direction maps etc., little modeling work has concentrated on functional aspects of topography.

A difficult and unsolved problem in modeling the visual system and building computer vision systems is the *invariance problem*: How is it possible to recognize objects despite their myriad of ways of appearing on the retina? Many models on invariant recognition in neuroscience are based on the idea of extracting features that yield responses that are insensitive to the invariance transformations desired. Often, this idea is implemented by a hierarchy of feature detectors alternating with pooling stages [2]. All these approaches have in common that they throw away the information, how the invariant representation came about, i.e. in particular where the information was pooled from. Resolving ambiguities, however, needs the interaction of higher level representations with lower, less invariant representations, as can be seen for e.g. segmentation or figure ground segmentation. Therefore, it is necessary that the processes that lead to the invariant responses are modeled explicitly, like e.g. in Dynamic Link Matching [3]. However, these correspondence-based approaches leave the organization of the representation in memory open. Simple bilinear models have been shown to be

able to organize transformations and the representations at the same time [4], usually employing some form of slowness.

In this work, we develop a bilinear model with slowness in section 2 and show in section 3 that the proposed framework is able to organize topographic representations of the input, i.e. maps of different kind, that are invariant to the simple transformations the network has been trained with. The model therefore builds “semantically ordered” invariant representations, while at the same time explicitly representing the underlying transformations that are necessary to yield invariance.

2 The Bilinear Topographic Model

We now define the bilinear generative model. Here, the input data x_i is assumed to be generated by a bilinear transformation of the latent variables and additive noise:

$$x_i = \sum_j^J \sum_k^K g_{ijk} y_j h_k + \eta_i, \quad (1)$$

where we assume η_i to be uncorrelated Gaussian noise with variance σ_i^2 , i.e. $\sigma_i = \sigma$, $\forall i$. The latent variables y_j and h_k can be identified with the firing rates of cortical cells. The generative distribution of the generative model of equation 1 is:

$$p(\mathbf{x}|G, \mathbf{y}, \mathbf{h}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left(-\frac{\sum_i \left(x_i - \sum_j^J \sum_k^K g_{ijk} y_j h_k\right)^2}{2\sigma^2}\right), \quad (2)$$

where N indicates the dimensionality of the input vector \mathbf{x} .

2.1 Topography

We model the emergence of topography following an idea discovered by Hyvärinen and Hoyer (see e.g. [5]): although the prior distributions over causes in standard linear models, like sparse coding or ICA, bias the responses of the latent variables to be mutually independent, the responses to natural images are not fully independent. This is because natural images have a too complex statistical structure to be transformed to fully independent causes by linear models. If we now assume the cells coding for the latents y_j to project to another layer of cells z_l , we can however make use of the Central Limit Theorem (CLT) to order the y_j according to their residual statistical dependencies, simply by forcing the next-layer cells z_l to be non-Gaussian. This is because if the cells y_j that project to cell z_l were independent, the response of z_l would tend to be more Gaussian. The resulting bilinear model, including this additional layer of units, is illustrated in figure 1 left. The responses of the highest cells in the model are modeled similar to the standard energy model of complex cells: $z_l = \sum_j \Gamma(l, j) y_j^2$, i.e. the

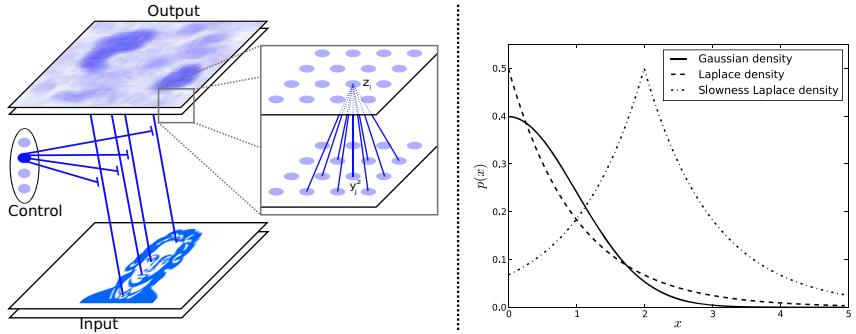


Fig. 1. Left: The bilinear generative model consists of three different groups of units: Feature units y_j , local pooling units z_l and control units h_k . Right: Normalized probability density functions. A Laplacian prior pdf (dashed line), i.e. $p(x) \propto \exp(-|x|)$, deviates from a Gaussian pdf (solid line), i.e. $p(x) \propto \exp\left(-\frac{x^2}{2}\right)$, in having higher probabilities for small as well as large values of x . Moving the mean μ of the Laplacian, i.e. $p(x) \propto \exp(-|x - \mu|)$, can be used to implement slowness or top-down information (e.g. for matching) in the model.

response of the cell z_l is given by pooling over the squared outputs of nearby cells y_j . $\Gamma(l, j)$ defines the neighborhood function and is set to 1 if y_j is in the vicinity of z_l and otherwise to 0.

We do not model the units z_l explicitly, but instead employ an according prior for y_j . As the resulting prior cannot be given in closed form under this missing variables model, a lower bound approximation of the prior has been derived in [5]:

$$\tilde{p}(\mathbf{y}) = \prod_i \exp \left(s \left(\sum_j \Gamma(i, j) y_j^2 \right) \right) \quad (3)$$

It is well known that the precise form of the sparsity forcing function s is not important for the results, as long as the overall shape of the function is correct. We use the sparsity forcing function s as suggested in [5]: $s(\xi) = -\alpha \sqrt{(\xi + \epsilon)}$, where ϵ guarantees numeric stability and is set to $\epsilon = 0.001$ for all our simulations. $\alpha = 2.0$ is an unimportant parameter as it can be absorbed in the learning rate. Note that for $\Gamma(i, j) = \delta_{ij}$ and $\epsilon = 0$ the resulting prior becomes the standard Laplacian distribution (see figure 1, right):

$$p(\mathbf{h}) = \prod_k \exp(-\alpha|h_k|), \quad (4)$$

which we used as a prior for the latent variables \mathbf{h} .

2.2 Slowness

A central goal of the proposed model is to build invariant topographic representations of the input data. Slowness has been proposed to play a central role in

unsupervised learning of invariant responses (e.g. [6]) and its rationale is that although the input data changes fast with time, the actual causes (e.g. objects) tend to change slow with time. Hence, if neurons try to code with slowly changing responses they are likely to catch invariant underlying causes. Slow Feature Analysis (SFA) [6] has been shown to be equivalent with probabilistic learning in a linear Gaussian model with an independent Markovian prior [7]. Similarly, we formalize this idea in our probabilistic model, by shifting the topographic prior probability around the latest responses, thus increasing the probability of the estimates to lie close to the latest values (see for an illustration Figure 11, right, the shifted Laplacian distribution):

$$p(\mathbf{y}) = \prod_i \exp \left(s \left(\sum_j \Gamma(i, j) \left(\frac{y_j - \tilde{y}_j}{\sigma_s} \right)^2 \right) \right) \quad (5)$$

where \tilde{y}_j is the output of the cell in the output field in the last time-step and σ_s parameterizes how much the estimated latent variable y_j is allowed to deviate from the latest response. For the simulations we set $\sigma_s = 1$. The simulations performed were not very sensitive to this parameter. Note that slowness for the latents \mathbf{y} is the main ingredient that breaks the symmetry of \mathbf{y} and \mathbf{h} and yields invariant responses in \mathbf{y} , while \mathbf{h} controls the transformations.

2.3 Dynamics and Learning Rule

After having defined the probabilistic model, we shall now derive the dynamics for the latent variables as well as the learning rule. Both can be derived by noting that the input statistics should be as close to the sample statistics from the generative model as possible. This density estimation procedure corresponds to maximizing the average log likelihood $\langle \log p(G|\mathbf{x}) \rangle$, where the marginal distribution is given by marginalizing over both \mathbf{y} and \mathbf{h} :

$$p(\mathbf{x}|G) = \int p(\mathbf{x}|G, \mathbf{y}, \mathbf{h}) p(\mathbf{y}) p(\mathbf{h}) d\mathbf{y} d\mathbf{h} \quad (6)$$

Unfortunately, the appearing integral in equation 6 is intractable. However, a common approximation is to replace the integral by an evaluation at the maximum a posteriori value (e.g. [8]):

$$(\hat{\mathbf{y}}, \hat{\mathbf{h}}) = \arg \max_{\mathbf{y}, \mathbf{h}} p(\mathbf{y}, \mathbf{h} | \mathbf{x}, G), \quad (7)$$

thus, ignoring the volume around the maximum.

In general, maximizing the average log likelihood is equivalent to minimizing the estimate of code length:

$$\mathcal{L} = -\log(p(\mathbf{x}|G, \mathbf{y}, \mathbf{h}) p(\mathbf{y}) p(\mathbf{h})). \quad (8)$$

By substituting the generative distribution, equation 2, and the prior distributions over the causes, equations 4 and 5, we arrive at the objective function of the model:

$$\begin{aligned} \mathcal{L} = & \sum_i \left(x_i - \sum_j \sum_k g_{ijk} y_j h_k \right)^2 \\ & - \frac{\lambda_y}{2} \sum_i s \left(\sum_j \Gamma(i, j) \left(\frac{y_j - \hat{y}_j}{\sigma_s} \right)^2 \right) - \frac{\lambda_h}{2} \sum_k s(h_k^2). \end{aligned} \quad (9)$$

The gradients for estimation of the latent variables are:

$$\begin{aligned} \frac{d}{dy_j} \mathcal{L} = & -2 \sum_i r_i \sum_k g_{ijk} h_k \\ & - \lambda_y \left(\frac{y_j - \hat{y}_j}{\sigma_s} \right) \sum_i s' \left(\sum_{j'} \Gamma(i, j') \left(\frac{y_{j'} - \hat{y}_{j'}}{\sigma_s} \right)^2 \right) \Gamma(i, j), \end{aligned} \quad (10a)$$

$$\frac{d}{dh_k} \mathcal{L} = -2 \sum_i r_i \sum_j g_{ijk} y_j - \lambda_h h_i s'(h_i^2), \quad (10b)$$

where $s'(\xi)$ denotes the derivative of the sparsity function s and we defined the residual image $r_i = x_i - \sum_j \sum_k g_{ijk} y_j h_k$. For the simulations, we used nonlinear conjugate (Polak-Ribière) gradient descent, using the log likelihood as defined in equation 9 and its gradient (equations 10) to estimate the latent variables \hat{y} and \hat{h} .

After the estimation of the latent variables, a simple generalized bilinear Hebbian learning step on the generative weights follows:

$$\tilde{g}_{ijk}(t) = g_{ijk}(t) + \eta_g \frac{d}{dg_{ijk}} \mathcal{L}, \quad \text{with} \quad \frac{d}{dg_{ijk}} \mathcal{L} = 2r_i \hat{y}_j \hat{h}_k, \quad (11)$$

which is applied with a fixed learning rate $\eta_g = 0.005$. After each learning step the weights get L2 normalized:

$$g_{ijk}(t+1) = \frac{\tilde{g}_{ijk}}{\sqrt{(\sum_i \tilde{g}_{ijk}^2)}}. \quad (12)$$

3 Simulations

3.1 Natural Inputs

We applied the proposed bilinear model to natural image data recorded by van Hateren [9]. Patches of size 32x32 were extracted at random positions from the images for the single control unit simulations ($\dim(\mathbf{h}) = 1$) and patches of size 16x16 for the two control unit case. Figure 2a shows a randomly selected patch. A common procedure in simplifying the estimation of higher-order models is to whiten the input data (e.g. in ICA). This means that second-order correlations are removed and variances of individual responses are normalized to one, to help

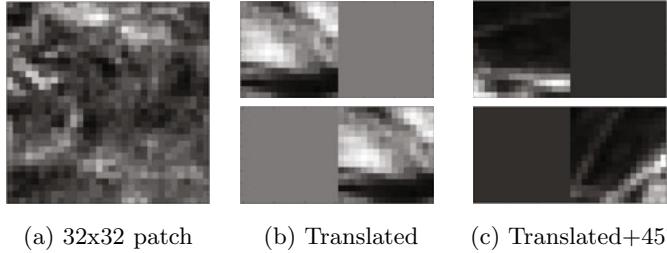


Fig. 2. Subfigure a) shows a natural image patch of size 32x32 used for the single control unit simulations. For two control units 16x16 sized natural inputs were placed in succession in a 32x16 field, see subfigures b) and c).

the learning method to focus on higher-order dependencies. For the illustrations of the generative weights after learning, they are projected back to the input space using the inverse whitening matrix.

In case of more control units we placed the randomly extracted image patches at different positions in a bigger input field. To guarantee independent transformations underlying the input data we placed the patches at disjoint positions. Figure 2b shows a pair of such inputs which are shown in succession, yet in random order, to the model and comply to a translation. Analogously, different transformations have been used, e.g. a translation and rotation with 45° in figure 2c. The presence of a single patch in a bigger field of vanishing activity can be interpreted as a schematic of early visual attention processing [10], which has been shown to suppress activity peripheral to the representation of a stimulus.

3.2 Parameters

30000 randomly selected input patches were extracted as described in section 3.1 and each input was reduced to the 100 or 200 dimensions with highest variance using PCA, for the 32x32 or the 16x16 inputs, respectively. Output fields were of size 24x24 for the single and 10x10 for the two control unit case. The sparseness parameters were set symmetrically to $\lambda_y = \lambda_h = 0.1$, i.e. \mathbf{h} and \mathbf{y} differ only in dimensionality and the estimation for \mathbf{y} includes slowness. For their estimation, all components from \mathbf{y} and \mathbf{h} were initialized independently to a value drawn from a standard Gaussian distribution with variance 0.1. Although we also tried bigger pooling sizes like 5x5 for $\Gamma(i, j)$, for all simulations shown the pooling size was a 3x3 neighborhood.

3.3 Results

In general, the learned generative fields are localized bandpass filters and are shown for the example of a single control unit in figure 4a. For a single control unit the model essentially reduces to a linear one, with a single modulatory interaction with the control unit activity h that can rescale and/or invert the activities of all feature coding units \mathbf{y} . The resulting basis functions are similar

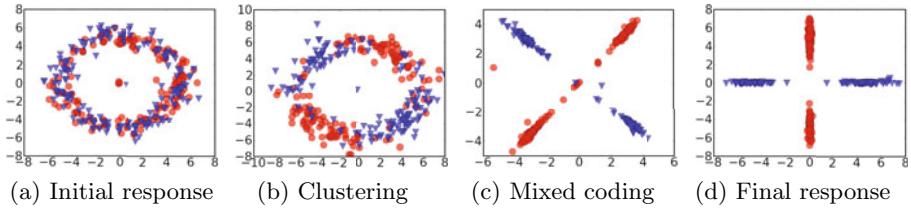


Fig. 3. Responses of the two control units to the two classes of transformed input patterns. In each subplot the last 150 estimations of h_2 (y-axis) are plotted against the responses of h_1 (x-axis). Red circles denote \mathbf{h} estimates inferred from input patches from the left in the input field, while blue upside down triangles denote estimates from the right. a) shows the initial response of the control units, while in b) at $t = 5208$ relatively short learning leads to clustering, c) at $t = 11444$ shows shrinkage to one-dimensional subspaces and d) shows the final result with sparsification of the responses.

to the ones learned in ICA or sparse coding, with the additional property of being arranged topographically according to position, orientation and frequency, including clusters of low wave numbers and discontinuities in orientation, similar to pinwheels in cortex. As for TICA [5], the results resemble the cortical organization of receptive fields in V1.

For the bilinear case of two control units, we shall first analyze the responses of the control units on the inputs. Figure 3 shows the development of the response of \mathbf{h} on the inputs. For a given time the responses of the last 150 estimations of h_2 are plotted against the responses of h_1 . Red circles denote \mathbf{h} estimates inferred from input patches from the left in the input field, while blue upside down triangles denote estimates from the right. It comes at no surprise, that the initial responses of the control units are circular symmetric for both inputs, as can be seen from figure 3a. This is clear, as all generative fields are random and the inputs are normalized. Learning clusters the population code for the different inputs, i.e. specific combinations of control unit activities code for either the left or the right input, see figure 3b. As learning proceeds, figure 3c, these clusters lie on one dimensional subspaces that are mutually orthogonal, thus reflecting a good code for the independence in the inputs. Finally, sparseness forces the one dimensional subspaces to lie on the coordinate axes themselves, because this is the most sparse code possible with only one unit coding for each input. Thus the final code after generative field organization effectively resembles a Winner-take-All response, if the input is chosen accordingly, yet can be a population response, if necessary.

The final generative fields can be visualized given an activity code for the control units. Figure 4b shows the resulting linear generative fields given $\mathbf{h} = (1, 0)$ and $\mathbf{h} = (0, 1)$, top and bottom, respectively. Though not as nicely localized as for the simpler linear case without slowness, the generative fields are clearly oriented band-pass filters and some also show good localization. Significant deviations from zero are only seen on one side of the fields, the side where the corresponding control unit got specialized in. We used this fact in the other sub-

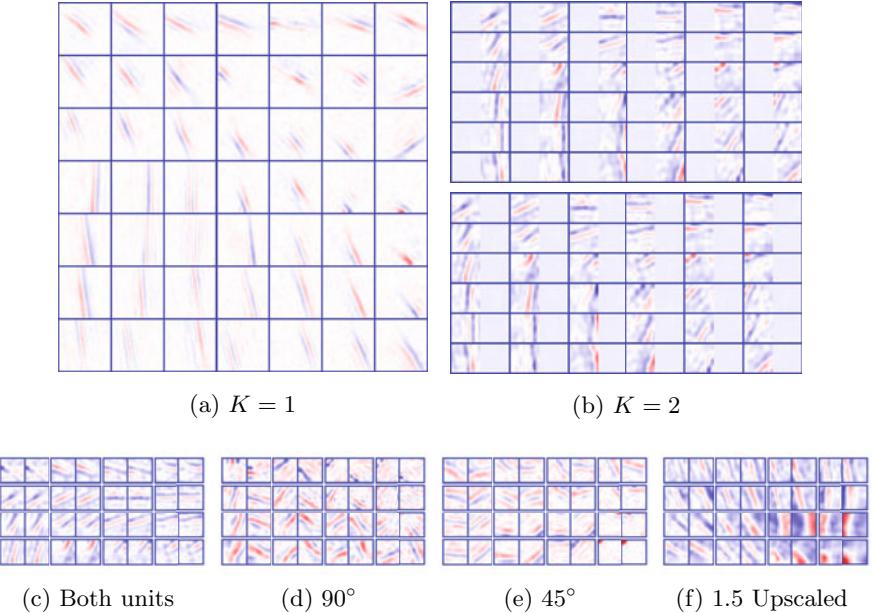


Fig. 4. a) Generative bilinear weights without slowness for a single control unit. b) Generative field for two control units given control unit activity responses of $\mathbf{h} = (1, 0)$ (top) and $\mathbf{h} = (0, 1)$ (bottom). c) shows that the non-zero parts of the generative fields are practically identical for both control unit activities, despite being translated. Similar, orientations around d) 90°, e) 45° and f) 1.5x upscaling yields accordingly transformed generative fields.

figures of figure 4, where only the non-zero parts of each given control unit field is plotted, side by side with the other control unit field, to simplify comparison of the generative fields. Figure 4c shows that slowness practically yielded identical generative fields for the two control units, despite the inputs being translated. Similarly, figures 4d and 4e show that this result generalizes to orientations of the input stimuli around 90° and 45° degrees, respectively. The same holds, if the inputs comply to relative scaling of 1.5, see figure 4f.

Parameters extracted from a Gabor-fitting analysis can be used to analyze the topographic continuity of the resulting generative fields with respect to the extracted parameters of the Gabor functions. For the case of a single control unit, maps extracted in this way are shown in figures 5a to 5d. The first map, figure 5a shows that neighboring output units, have similar input positions (the input position is coded with the color-code given in figure 5i), the map is continuous and mostly smooth with occasional discontinuities. This result corresponds to the finding of “retinotopy” in visual cortex. Figure 5b shows that the extracted orientations are even smoother, while for frequencies, figure 5c, clusters can be

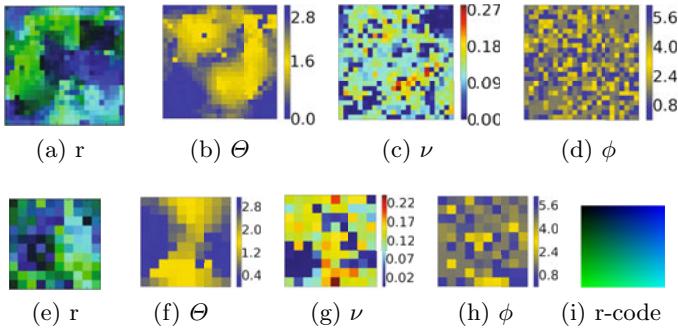


Fig. 5. Visualization of the topographic maps with respect to position (r), orientation (Θ), frequency (ν) and phase (ϕ) of the generative fields extracted by fitting Gabor functions. Subplot (a)–(d) show results for a single control unit while (e)–(h) shows results for two control units. For the position plots the color-coding in the input is given in subplot (i). Orientation is color-coded as given by the color-bar in $[0, \pi]$ and phases are given accordingly in $[0, 2\pi]$.

seen, yet the map is noisy. Similar to simple cells in cortex, phases show no apparent topography, see figure 5d.

Figures 5e to 5h show the analogue results for the bilinear case and given $\mathbf{h} = (1, 0)$. The Gabor fits were restricted to the areas in the generative field that was significantly non-zero. Qualitatively the maps are very similar to the single control unit case. Orientation continuity and frequency topography is more pronounced, while position continuity is slightly more noisy.

4 Discussion

We introduced a topographic bilinear model for the purpose of building invariant representations. Like in [5] nearby output cell generative fields are organized to maximize the mutual information of their output activities. A Gabor analysis showed that topography emerges with respect to several parameters: location, orientation and frequency. In the case of two control units, their generative fields were shown to comply to the relative transformation of the input data.

Several models in the recent literature attempted to separate features and transformations using bilinear models (e.g. [4]). All these models have in common that they do infer their latent variables on at least two successive images or a whole batch of images (see however [11]). In contrast, our model uses a single input image to infer both the feature and control unit activities. This is achieved by implementing slowness in the prior of the feature coding units, to break the symmetry with the control units, and simplified by the assumption of an attention signal in the input. Simple disjoint inputs were chosen for analysis purposes and future work will show how the model generalizes to more complex situations.

It is widely accepted in the neuroscientific community, that the cortex is organized hierarchically, and it would therefore be desirable to exploit this organization. However, standard linear methods in representational learning, like sparse coding [8] or independent component analysis, do not benefit from hierarchies, as representations would not change significantly in the hierarchy. The topographic bilinear model of this paper, on the other hand, is a well suited module in a hierarchy, due to its inherent non-linearity and vicinity coding of related information. Further, the shifting prior currently used for implementing slowness, can be used for integration of top-down information, i.e. information from presumably higher cortical areas.

Acknowledgement. Supported by EU project “SECO”, and the Hertie Foundation.

References

1. Willshaw, D.J., von der Malsburg, C.: How patterned neural connections can be set up by self-organization. *Proc. R. Soc. Lond. B. Biol. Sci.* 194(1117), 431–445 (1976)
2. Fukushima, K.: Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* 36(4), 193–202 (1980)
3. Wolfrum, P., Wolff, C., Lücke, J., von der Malsburg, C.: A recurrent dynamic model for correspondence-based face recognition. *J. Vis.* 8(7), 34.1–34.18 (2008)
4. Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. *Neural Computation* 12(6), 1247–1283 (2000)
5. Hyvärinen, A., Hoyer, P.O., Inki, M.: Topographic independent component analysis. *Neural Comput.* 13(7), 1527–1558 (2001)
6. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: unsupervised learning of invariances. *Neural Computation* 14(4), 715–770 (2002)
7. Turner, R., Sahani, M.: A maximum-likelihood interpretation for slow feature analysis. *Neural Comput.* 19(4), 1022–1038 (2007)
8. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381(6583), 607–609 (1996)
9. van Hateren, J.H., van der Schaaf, A.: Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc. Biol. Sci.* 265(1394), 359–366 (1998)
10. Hopf, J.M., Boehler, C.N., Luck, S.J., Tsotsos, J.K., Heinze, H.J., Schoenfeld, M.A.: Direct neurophysiological evidence for spatial suppression surrounding the focus of attention in vision. *Proc. Natl. Acad. Sci. USA* 103(4), 1053–1058 (2006)
11. Berkes, P., Turner, R.E., Sahani, M.: A structured model of video reproduces primary visual cortical organisation. *PLoS Comput. Biol.* 5(9), e1000495 (2009)

Accelerating Large-Scale Convolutional Neural Networks with Parallel Graphics Multiprocessors

Dominik Scherer, Hannes Schulz, and Sven Behnke

University of Bonn, Institute of Computer Science VI,
Autonomous Intelligent Systems Group, Römerstr. 164, 53117 Bonn, Germany
`{scherer,schulz}@ais.uni-bonn.de, behnke@cs.uni-bonn.de`
<http://www.ais.uni-bonn.de>

Abstract. Training convolutional neural networks (CNNs) on large sets of high-resolution images is too computationally intense to be performed on commodity CPUs. Such architectures, however, achieve state-of-the-art results on low-resolution machine vision tasks such as recognition of handwritten characters. We have adapted the inherent multi-level parallelism of CNNs for Nvidia’s CUDA GPU architecture to accelerate the training by two orders of magnitude. This dramatic speedup permits to apply CNN architectures to pattern recognition tasks on datasets with high-resolution natural images.

1 Introduction

Biologically-inspired convolutional neural networks (CNNs) have achieved state-of-the-art results for the recognition of handwritten digits [5,11] and for the detection of faces [1,8]. However, since gradient-based learning of CNNs is computationally intense, it would require weeks to train large-scale CNNs on commodity processors. It therefore remains a largely unexplored question whether CNNs are viable to categorize objects in high-resolution camera images.

In the 1990ies similar incentives to parallelize neuroapplications led to the development of both general-purpose and special-purpose neurohardware. Popular data-parallel techniques primarily relied on the parallel execution of nodes for a single input pattern [13]. Early approaches to implement CNNs on parallel graphics hardware [2] yielded speedups of up to 4.11 \times . More recently, multilayer perceptrons [3], locally-connected neural networks [12] and deep belief networks [9] have been adapted to GPUs with speedups by two orders of magnitude.

Modern graphics cards consist of several hundred parallel processing cores. Nvidia’s scalable CUDA framework [7] (Compute Unified Device Architecture) makes it possible to harness their computational power of several hundreds of GigaFLOPS (floating point operations per second) without requiring to learn a specialized programming language. However, in order to accelerate an application, hardware-imposed memory access restrictions must be considered to efficiently exploit the fast on-chip shared memory. Both general neural networks and convolution operations are inherently parallel. Thus, CNNs, which combine both, are a particularly promising candidate for a parallel implementation.

To account for the peculiarities of such CUDA-capable devices, we had to slightly divert from common CNN architectures. However, our CNN model for CUDA’s parallel hardware architecture is sufficiently flexible for a large range of machine vision tasks. We describe a fast parallel implementation of the backpropagation algorithm to train this network on GPU hardware. The program scales well on an arbitrary number of processors, and employs a circular buffer strategy to minimize data transfers from the device memory to the on-chip shared memory. This implementation achieves a speed-up factor between 95 and 115 over a serial, single-core CPU implementation and scales well with both the network and input size.

2 Massively Parallel Computing on GPUs

Modern graphics processing units (GPUs) have evolved from pure rendering machines into massively parallel general purpose processors. Recently, they exceeded 1 TeraFLOPS [7], outrunning the computational power of commodity CPUs by two orders of magnitude. GPUs employ the fundamentally different SPMD (Single Program, Multiple Data) architecture and are specialized for intense, highly parallel computations. More transistors are devoted to data processing rather than to data caching and control flow. The CUDA framework introduced by Nvidia allows the development of parallel applications through “C for CUDA”, an extension of the C programming language.

The CUDA programming model considers a graphics card (*device*) as a physically separate co-processor to the CPU (*host*). Computations on the GPU are initiated through *kernel functions* which essentially are C-functions being executed in N parallel *threads*. Semantically, threads are organized in 1-, 2- or 3-dimensional groups of up to 512 threads, called *blocks*, as shown in Figure 1a. Each block is scheduled to run separately from all others on one multiprocessor. Blocks can be executed in arbitrary order – simultaneously or in sequence, depending on the system’s resources. However, this scalability comes at the expense of restricting the communication between threads.

Threads have access to several memory spaces, as illustrated in Figure 1a. Each thread has a small private *local memory* space. In addition, all threads within the same block can communicate with each other through a low-latency *shared memory*. The much larger *global memory* has a higher latency and can be accessed by the CPU, thus being the only communication channel between CPU and GPU. The *GeForce GTX 285* graphics card running in our system consists of 30 multiprocessors with 8 stream processors each, resulting in a total of 240 cores and yielding a maximum theoretical speed of 1,063 GFLOPS. Each multiprocessor contains 16 KB of on-chip shared memory as well as 16,384 registers. The GPU-wide 1024 MB of global memory can be accessed with a maximum bandwidth of 159 GB per second.

In order to run hundreds of threads concurrently, the multiprocessors employ an architecture called SIMT (Single Instruction, Multiple Threads), visualized in Figure 1b. The SIMT unit of a multiprocessor creates, schedules and executes

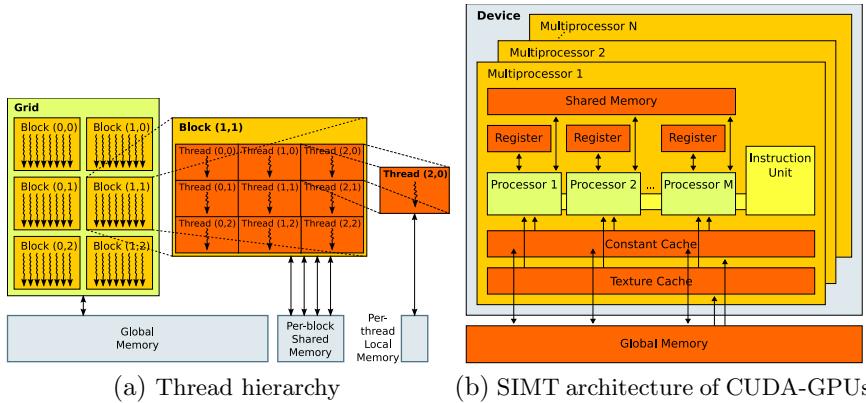


Fig. 1. (a) Grids are subdivided into blocks, which consist of many parallel threads. Threads of the same block can cooperate through shared memory. (b) A device consists of several multiprocessors, each with their own memory spaces. (adapted from [7]).

groups of 32 consecutive parallel threads. The highest efficiency can be achieved if all threads execute the same instruction path. Memory accesses from different threads can be *coalesced* into one memory transaction if consecutive threads access data elements from the same memory segment. Following such specific access patterns can dramatically improve the memory utilization and is essential for optimizing the performance of an application. Despite such optimizations, the CUDA framework is best suited for applications with a high *arithmetic density*, that is, where the number of memory transactions is small compared to the number of arithmetic operations.

3 CNN Architectures for SIMT Processors

The main concept of Convolutional Neural Networks (CNNs) is to extract local features at a high resolution and to successively combine these translation-invariant features into more complex features at lower resolutions. The loss of spatial information is compensated for by an increasing number of feature maps in the higher layers. Usually, CNNs consist of two altering types of layers, convolutional layers and subsampling layers. Each convolutional layer performs a discrete folding operation of its source image with a filter kernel. The subsampling layers reduce the size of the input by averaging neighboring pixels. Our architecture – shown in Figure 2 – diverges from this typical model because both the convolution and subsampling operations are performed in one step.

This modification was necessary due to the limited memory resources of the GPU hardware: During the backpropagation step it is required to store both the activities and the error signal for each feature map and each pattern. When combining both processing steps, the memory footprint of each feature map can thus be reduced by a factor of four when 2×2 subsampling is applied.

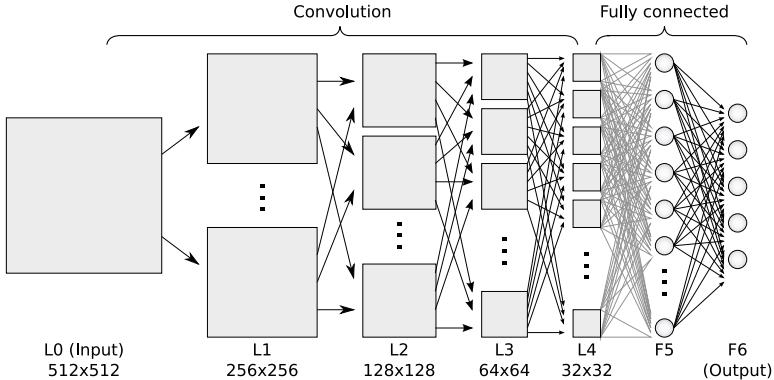


Fig. 2. Architecture of our CNN. The lower convolutional layers are locally connected with one convolutional kernel for each connection. The last convolutional layer L4 is followed by two fully connected layers.

The activity $a_j^{(l)}(x, y)$ of a neuron at position (x, y) in feature map j of layer l solely depends on the activities of neurons in a local receptive field of all feature maps from the preceding layer $l - 1$. Each filter not only consists of its elements $w_{ij}(u, v)$ at position (u, v) , but also of a bias weight b_{ij} . The net input $net_j(x, y)$ of feature map j is calculated over all source maps I :

$$net_j(x, y) = \sum_{i=0}^I \left(\left(\sum_{(u,v)} w_{ij}(u, v) \cdot a_i(2x + u, 2y + v) \right) + b_{ij} \right). \quad (1)$$

A non-linear sigmoid function – here we use hyperbolic tangent – is applied to determine the activity $a_j(x, y)$ of each neuron:

$$a_j(x, y) = f_{act}(net_j(x, y)) = \tanh(net_j(x, y)). \quad (2)$$

For our implementation, we used 8×8 kernels which are overlapping by 75% in each direction. The receptive field of a neuron increases exponentially on higher layers. Each neuron in layer L4 is influenced by an area of 106×106 pixels of the input image. A more technical advantage of this choice is that the $8 \times 8 = 64$ filter elements can be processed by 64 concurrent threads. In the CUDA framework, those 64 threads are coalesced into two *warps* of 32 concurrent threads each.

The filter weights are adapted with the gradient descent algorithm *backpropagation of error*. For one particular training pattern, the error signal δ_k of a neuron k in the output layer is calculated from the desired output t_k and the actual output o_k :

$$\delta_k = f'_{act}(net_k) \cdot (t_k - o_k). \quad (3)$$

For a fully connected hidden neuron j , this error is propagated backwards with

$$\delta_j = f'_{act}(net_j) \cdot \sum_k \delta_k w_{jk}. \quad (4)$$

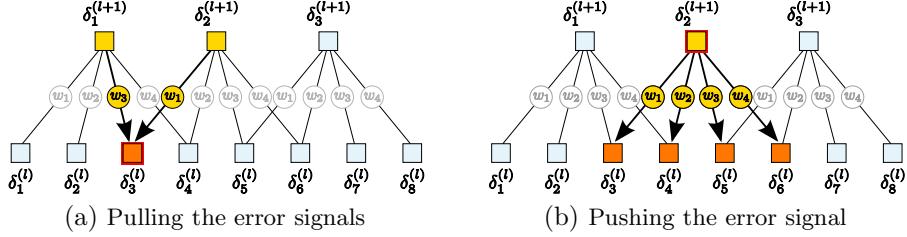


Fig. 3. Simplified example of the backpropagation pass for a one-dimensional case.

(a) When the error signals are pulled by a neuron of the lower layer, discontinuous weights and neurons of the higher layer are involved. (b) When pushing the error signals, a neuron from the higher layer iterates the weights and accesses the error signals of continuous neurons.

For the convolutional layers it is error-prone to calculate the error due to the dimension-reducing filter kernels: Because of the subsampling operation, the index arithmetics for the error signal computations are complex. Therefore, we are employing a technique that Simard *et al.* [11] call “pushing” the error signals, as opposed to “pulling”. Figure 3 depicts the difference between those operations for a 1D case. When the error is “pulled” by a neuron from the lower layer, it is tedious to determine the indices of the weights and neurons involved. On the other hand, “pushing” the error signal can be considered as the inversion of the forward pass, which enables us to use similar implementation techniques.

The convolutional kernel is applied at every position of a feature map, thus the weights are shared between several connections. The partial derivative of the error with respect to a weight $w_{jk}(u, v)$ is not only summed over all patterns P , but also over all positions (x, y) of the feature map:

$$\frac{\partial E}{\partial w_{jk}(u, v)} = \sum_{p=1}^P \sum_{(x,y)} \left(-\delta_k^{(l+1)}(x, y) \cdot a_j^{(l)}(2 \cdot x + u, 2 \cdot y + v) \right). \quad (5)$$

The weight update $\Delta w_{kj}(u, v)$ can then be calculated as

$$\Delta w_{ji}(u, v) = -\eta \cdot \frac{\partial E}{\partial w_{jk}(u, v)}, \quad (6)$$

with an adjustable learning rate of $\eta > 0$.

Determining a suitable learning rate can be difficult. Randomly initialized deep networks with more than three hidden layers often converge slowly and reach an inadequate local minimum [4]. During our experiments, it proved impossible to empirically choose a suitable learning rate for each layer. We therefore implemented the RPROP algorithm [10] which maintains an adaptive learning rate for each weight.

4 Implementation

To accelerate an application with the CUDA framework, it is necessary to split the problem into coarse-grained sub-problems which can be solved independently.

Each task is mapped to a multiprocessor. Within each block, this task is further divided into finer pieces that can be solved cooperatively in parallel by many threads. We decided to assign each training pattern to one block, which implies that a mini-batch learning scheme has to be applied. During one batch learning pass, every pattern uses the same network parameters and weights.

The number of training patterns to be processed in parallel is restricted by the limited amount of global memory. With 32-bit precision, a feature map of 512×512 pixels occupies 1 MB of memory. The activities of all layers and all feature maps are required in the training pass, hence they all contribute to the memory footprint of one training pattern. Depending on the number of feature maps per layer and the size of the feature maps, up to 10 MB might be required for each pattern. When using the GeForce 285 GTX this means that only a few dozen patterns can be processed concurrently. At a finer scale, i.e., within a block, we perform the convolution of one source feature map onto eight target feature maps simultaneously. With this setup, the activities of the source feature map need to be loaded only once for every eighth target feature map. This dramatically reduces the amount of memory transfers. To simplify the parallel implementation described here, we will only consider the case of 64×64 pixel source feature maps and 32×32 pixel target feature maps. It can be expanded for multiples of these sizes with little effort.

Because shared memory is extremely limited, it is critically important to reuse loaded data as often as possible. Even if the whole 16 KB of shared memory is used, it can only hold a small fraction of the source feature map. For this reason, we are utilizing the shared memory as a circular buffer which only holds a small region of the source feature map, as shown in Figure 4. During each iteration only two rows of this buffer need to be exchanged. A 70×8 window of the source feature map is maintained within shared memory. This area includes the 64 columns of the source map as well as a 3 pixel wide border on both sides. The 8×8 convolutional filter is applied at every other pixel, thus fitting exactly 32 times into this 70×8 window.

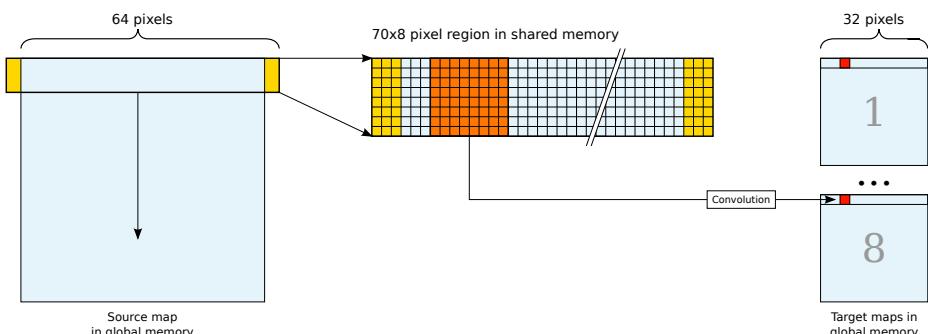


Fig. 4. Convolution operation. A region of the source map is loaded from global memory into the circular buffer in shared memory. Each thread performs a convolution on this section and subsequently writes the result into the target map in global memory.

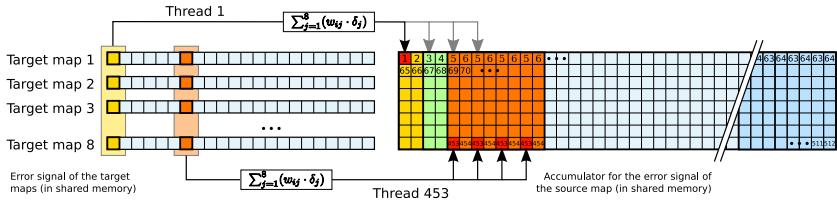


Fig. 5. Aggregation of the backpropagated error signals. Each thread first sums the partial error signal over one position in all eight target map rows. Then it writes the result to the accumulator in shared memory. As indicated by the thread numbers on the right, each thread of a warp writes to a different memory bank. Thus, bank conflicts are avoided. This step is repeated four times to cover each element of a filter.

The source feature map is being iterated from top to bottom. During each iteration, three steps are performed: (1) Loading two rows of 70 consecutive pixels of the source feature map from global into shared memory with a coalesced transfer operation. (2) Iterating through each filter element to compute the convolution. (3) Writing 32 consecutive pixels, each computed by continuous threads, back to global memory. Since this kernel calculates the output of a 32 pixel wide row for eight target feature maps, it uses 256 threads. The source image window only requires $70 \times 80 \times 4 = 2240$ Byte of shared memory. Therefore, it is theoretically possible to run seven thread blocks on one multiprocessor at once. In practice, only four blocks can be scheduled simultaneously because each thread requires at least 16 registers. Data transfers are minimized as each loaded pixel is reused 128 times (at 16 positions for each of the 8 target maps). The result of our optimizations is that the runtime is bounded by the maximum arithmetic performance of the hardware as opposed to being bounded by the memory bandwidth.

During the backpropagation step, two tasks are performed simultaneously: the weight gradients are calculated for each element of the eight convolutional filters and the error signal is accumulated on the lower layer. For both operations, the error signal of the higher layer is required. Hence, it is reasonable to reuse this data once it is loaded. We are again handling eight convolutional filters during one device function call: 512 threads can be employed for this, one for each of the eight 8×8 filters. Similar to the forward pass, we are using a circular buffer to load the activities of the source map $a_i^{(l)}$ from global memory into shared memory. An additional circular buffer is used as an accumulator for the backpropagated error signals $\delta_j^{(l+1)}(x, y)$ of the target maps.

One thread runs through all 32 pixels of the row for one specific feature map. It multiplies the error signal with the activity of the source map pixel corresponding to its weight position and accumulates this value in a register. Before the kernel function terminates, the final value for the weight gradient is written to global memory. Access to the error signals of the target maps is coalesced because all 32 pixels of a row are loaded simultaneously. Similarly, loading the error signals from shared memory is optimal. All of the 64 threads (two warps) of a specific filter access the same memory location, which enables

the CUDA driver to broadcast this value to all threads. Write-access to global memory is coalesced as well, because all 64 elements of a filter are stored in succession.

As described in Section 3, we invert the forward pass to propagate the error signal $\delta_j^{(l+1)}(x, y)$ from the target layer $l + 1$ back to the source layer l . The first step is to accumulate the partial error signal in a circular buffer in shared memory. Once these values are pushed out of the circular buffer, they are added to the partial error signals in a temporary memory space in global memory.

This part of the algorithm reuses the error signals of the target maps which already reside in shared memory from the weight gradient computation. In order to avoid bank conflicts when accessing shared memory, our thread indexing scheme is not straight-forward, as illustrated in Figure 5. Each thread calculates and sums the partial error signal at a specific position for all eight target feature map rows. This sum is then accumulated in the shared memory circular buffer. As indicated in the figure, consecutive threads write to consecutive memory elements, thus avoiding bank conflicts. This step is repeated four times.

The ratio between memory transactions and arithmetic operations for the backpropagation pass is high because the error signals of the target maps are reused 64 times.

5 Evaluation

To verify the correctness of our implementation, we evaluated our CNN architecture on the normalized-uniform NORB dataset [6]. This dataset consists of 24,300 binocular grayscale training images and the same number of testing images, depicting toy figures of five categories. After training on the raw pixel data for 360 epochs (291,000 mini-batch updates), an error rate of 8.6% on the test set was achieved. To evaluate our architecture on the MNIST dataset of handwritten digits [5], we scaled the input images from 28×28 pixels to 256×256 , because our implementation requires large input patterns. After 10 epochs of backpropagation, this network achieved a test error rate of 1.38%.

The speed of our implementation was benchmarked on a system with an *Intel Core i7 940* (2.93 GHz) CPU and a *Nvidia GeForce GTX 285*. For a comparison of the runtime, we implemented a functionally equivalent single-threaded, sequential CPU version. The most critical components of the parallel implementation are the forward pass and the the backpropagation of error. For both, several patterns are processed in parallel, as described in Section 4. Thus, it is to be expected that a larger speedup can be achieved for an increasing number of patterns. Figure 6 shows the runtime of the setup described above as a function of the number of patterns processed in parallel. For any value between 1 and 30 patterns, the runtime remains almost constant, with a sudden jump at 31 patterns which is repeated every 30 patterns. Thus, as shown in Figure 7, the highest speedup was achieved when the number of patterns is a multiple of 30. This discontinuity is owed to the hardware: the GTX 285 GPU consists of 30 multiprocessors, each processing one pattern. If more then 30 patterns are

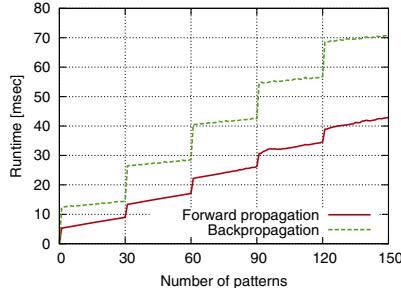


Fig. 6. Runtime of the forward and backpropagation passes as a function of the number of training patterns

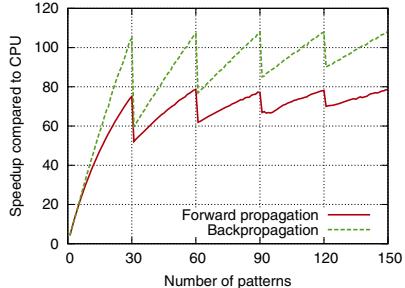


Fig. 7. Speedup of the parallel GPU implementation compared to a serial CPU version

scheduled, the remaining patterns are queued until another multiprocessor has finished.

When choosing a suitable number of patterns, the forward pass can be accelerated by a factor of 80. A maximum speedup factor of 110 is achieved for the backpropagation pass. In addition to the forward and backward pass, memory transfers are necessary to copy the training patterns from host memory to device memory. The CPU version has the advantage of using hardly any explicit memory transfers and memory access is accelerated by caching. On the contrary, for parallel GPU implementations memory transactions are often a bottleneck. To ensure a realistic comparison it is therefore necessary to include memory transfers in the total runtime of our GPU implementation.

We timed the total runtime of both our serial and parallel implementations for networks with a varying number of feature maps on each layer and with varying input sizes. Despite memory transactions being included in the benchmark results in Table II speedups of two orders of magnitude are achieved. Thus, we conclude that data transfer times are negligible compared to forward and backward propagation. We can also deduce that neither network size, nor the size of the input has a significant influence on the speedup of our implementation. In

Table 1. Benchmarks for one epoch with 60 training patterns, including all memory transfers. The networks have two fully connected layers with 100 and 10 neurons.

Input Size	Feature maps	CPU [ms]	GPU [ms]	Speedup
256×256	1-8-8-8	14,045	141	99.8
256×256	2-8-16-32	44,242	383	115.7
256×256	4-16-64-64	278,010	2,583	107.6
512×512	1-8-8-8-8	53,495	561	95.4
512×512	2-8-16-32-64	225,110	2,045	110.1
512×512	4-8-32-64-64	545,803	5,143	106.1

all cases, the gradient descent mini-batch learning was accelerated by a factor ranging from 95 to 115.

6 Conclusion

Our work shows that current graphics cards programming frameworks with their hierarchy of threads are very well suited for a parallel implementation of CNNs. In comparison to a serial CPU implementation, we were able to significantly speed up the learning algorithm for a large convolutional neural network by a factor ranging from 95 to 115 on a single GPU. Until now, it was impossible to study deep CNNs with high-resolution input images due to the tremendous computational power required for their training. The aim of our future work is to extend this approach to systems with multiple GPUs and to apply large-scale CNNs to pattern recognition tasks using datasets with millions of high-resolution natural images.

References

1. Behnke, S.: Hierarchical Neural Networks for Image Interpretation. LNCS, vol. 2766. Springer, Heidelberg (2003)
2. Chellapilla, K., Puri, S., Simard, P.: High Performance Convolutional Neural Networks for Document Processing. In: Tenth International Workshop on Frontiers in Handwriting Recognition, La Baule, France, Université de Rennes (2006)
3. Lahabar, S., Agrawal, P., Narayanan, P.J.: High Performance Pattern Recognition on GPU. In: Proc. of National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (January 2008)
4. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring Strategies for Training Deep Neural Networks. Journal of Machine Learning Research (2009)
5. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
6. LeCun, Y., Huang, F., Bottou, L.: Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In: Proc. of CVPR 2004 (2004)
7. Nvidia Corporation. CUDA Programming Guide 3.0 (February 2010)
8. Osadchy, M., LeCun, Y., Miller, M.: Synergistic Face Detection and Pose Estimation with Energy-Based Models. Journal of ML Research (2007)
9. Raina, R., Madhavan, A., Ng, A.Y.: Large-scale deep unsupervised learning using graphics processors. In: ACM Intl. Conference Proceeding Series (2009)
10. Riedmiller, M., Braun, H.: RPROP – A fast adaptive learning algorithm. In: Proc. of the Int. Symposium on Computer and Information Science VII (1992)
11. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best Practice for Convolutional Neural Networks Applied to Visual Document Analysis. In: ICDAR (2003)
12. Uetz, R., Behnke, S.: Large-scale Object Recognition with CUDA-accelerated Hierarchical Neural Networks. In: Proc. of ICIS (2009)
13. Šerbedžija, N.: Simulating artificial neural networks on parallel architectures (1996)

Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition

Dominik Scherer, Andreas Müller*, and Sven Behnke

University of Bonn, Institute of Computer Science VI,
Autonomous Intelligent Systems Group, Römerstr. 164, 53117 Bonn, Germany
`{scherer,amueller}@ais.uni-bonn.de, behnke@cs.uni-bonn.de`
<http://www.ais.uni-bonn.de>

Abstract. A common practice to gain invariant features in object recognition models is to aggregate multiple low-level features over a small neighborhood. However, the differences between those models makes a comparison of the properties of different aggregation functions hard. Our aim is to gain insight into different functions by directly comparing them on a fixed architecture for several common object recognition tasks. Empirical results show that a maximum pooling operation significantly outperforms subsampling operations. Despite their shift-invariant properties, overlapping pooling windows are no significant improvement over non-overlapping pooling windows. By applying this knowledge, we achieve state-of-the-art error rates of 4.57% on the NORB normalized-uniform dataset and 5.6% on the NORB jittered-cluttered dataset.

1 Introduction

Many recent object recognition architectures are based on the model of the mammal visual cortex proposed by Hubel and Wiesel [4]. According to their findings, the visual area V1 consists of of *simple cells* and *complex cells*. While simple cells perform a feature extraction, complex cells combine several such local features from a small spatial neighborhood. It is assumed that spatial pooling is crucial to obtain translation-invariant features.

Supervised models based on those findings are the Neocognitron [6] and Convolutional Neural Networks (CNNs) [10]. Many recent state-of-the-art feature extractors employ similar aggregation techniques, including Histograms of Oriented Gradients (HOG) [3], SIFT descriptors [12], Gist features [22], and the HMAX model [20].

These models can be broadly distinguished by the operation that summarizes over a spatial neighborhood. Most earlier models perform a subsampling operation, where the average over all input values is propagated to the next layer. Such architectures include the Neocognitron, CNNs and the Neural Abstraction

* This work was supported in part by NRW State within the B-IT Research School.

Pyramid [2]. A different approach is to compute the maximum value in a neighborhood. This direction is taken by the HMAX model and some variants of CNNs.

While entire models have been extensively compared, there has been no research evaluating the choice of the aggregation function so far. The aim of our work is therefore to empirically determine which of the established aggregation functions is more suitable for vision tasks. Additionally, we investigate if ideas from signal processing, such as overlapping receptive fields and window functions can improve recognition performance.

2 Related Work

Many computer vision architectures inspired by studies of the primary visual cortex use a multi-stage processing of simple and complex cells. Simple cells perform feature detection at a high resolution. Translation-invariance and generalization is achieved by complex cells, which combine activations over a local neighborhood.

One of the earliest models employing this technique is the Neocognitron [6]. Here, each of the so-called C-cells receives excitatory input connections from feature extraction cells at slightly different positions. A C-cell becomes active if at least one of their inputs is active, thus tolerating slight deformations and transformations.

In Convolutional Neural Networks (CNNs), such as LeNet-5 [10], shift-invariance is achieved with subsampling layers. Neurons in these layers receive input from a small non-overlapping receptive field of the previous layer. Each neuron computes the sum of its inputs, multiplies it by a trainable coefficient, adds a trainable bias and passes the result through a non-linear transfer function. A similar computation is performed in the recurrent Neural Abstraction Pyramid [2]. More recently, the subsampling operation in CNNs has been replaced with a max pooling operation [18]. Here, only the maximum value within the receptive field is propagated to the next layer.

In the global scene description computed by the *Gist* model [22], the feature extractor is not trainable, but performs similar computations. Low-level center-surround features are computed from color and intensity channels at different scales and orientations. Subsequently, each channel is divided into a 4×4 grid of sub-regions. The 16-dimensional Gist feature vector of this channel is computed by averaging the values in each region.

The SIFT [12] (scale-invariant feature transform) keypoint descriptor is computed by sampling the dominant orientation and the gradient magnitude around the keypoint location. These values, weighted by a Gaussian window function, are then accumulated into arrays of 4×4 orientation histograms summarizing the content over 4×4 positions. Pyramids of such local orientation histograms computed on a dense, overlapping grid of image patches have proven to be well suited scene and object representations for recognition tasks [9]. Experiments with locally normalized Histogram of Oriented Gradients (HOG) descriptors

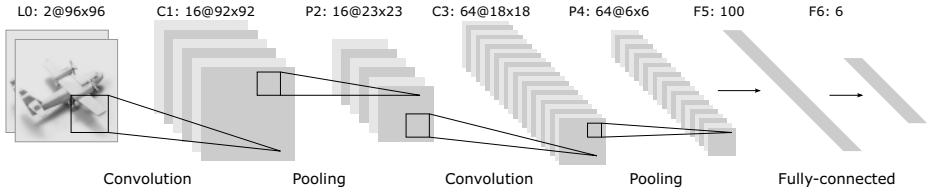


Fig. 1. Architecture of our CNN for NORB experiments, consisting of alternating convolutional and pooling layers. Pooling layers can implement either subsampling operations or max pooling.

have shown that fine-scale gradients and coarse-scale spatial binning yield good recognition performance for human detection [3].

Riesenhuber and Poggio originally proposed to use a maximum operation to model complex cells in 1999 [20]. In the HMAX model [21], each complex cell receives input from a distinct patch of a simple cell map. This approach was further refined by Mutch *et al.* [14], introducing sparsification and lateral inhibition. Here, max pooling is performed on overlapping patches with an overlap factor of two in a spatial neighborhood and across scales.

Even though all of these methods resemble the concept of simple and complex cells, they differ by the type of inputs, the method of feature extraction, the training algorithm, and the classifier used. Additionally, each method emulates complex cells slightly different. Due to those variations a thorough analysis how a particular choice of this component affects overall performance is impossible. In order to empirically compare the sole influence of different aggregation functions, we chose an otherwise fixed model architecture, described in the following section.

3 Model Architecture

This section describes the architecture of the feature extraction and classification system, as well as the training procedure used in our experiments. We chose to perform our evaluations within the framework of a Convolutional Neural Network (CNN). CNNs have achieved state-of-the-art results for the recognition of handwritten digits [23] and for the detection of faces [17]. They are deployed in commercial systems to read checks [10] and to obfuscate faces and license plates in Google StreetView [5].

3.1 Base Model

CNNs are representatives of the multi-stage Hubel-Wiesel architecture, which extract local features at a high resolution and successively combine these into more complex features at lower resolutions. The loss of spatial information is compensated by an increasing number of feature maps in the higher layers. CNNs consist of two altering kinds of layers: convolutional layers (C layers), which

resemble the *simple cells*, and pooling layers (P layers), which model the behavior of *complex cells*. Each convolutional layer performs a discrete 2D convolution operation on its source image with a filter kernel and applies a non-linear transfer function. The pooling layers reduce the size of the input by summarizing neurons from a small spatial neighborhood. Our choice of a CNN is largely motivated by the fact that the operation performed by pooling layers is easily interchangeable without modifications to the architecture. The general architecture is shown in Figure 1 and is identical to the models by LeCun *et al.* [1] and Ahmed *et al.* [2].

3.2 Convolutional Layers

Computations for the forward pass and the backpropagation in the convolutional layer follow the standard procedure in the literature, and have trainable filters and one trainable bias per feature map. A hyperbolic tangent function is applied to activations in this layer. Our experiments have shown that a sparse connection between feature maps does not improve recognition performance compared to fully connected feature maps as long as the number of parameters is equal. Thus, in a convolutional layer, each map is connected to all of its preceding feature maps.

3.3 Pooling Layers

The purpose of the pooling layers is to achieve spatial invariance by reducing the resolution of the feature maps. Each pooled feature map corresponds to one feature map of the previous layer. Their units combine the input from a small $n \times n$ patch of units, as indicated in Figure 1. This pooling window can be of arbitrary size, and windows can be overlapping.

We evaluate two different pooling operations: max pooling and subsampling. The subsampling function

$$a_j = \tanh(\beta \sum_{N \times N} a_i^{n \times n} + b) \quad (1)$$

takes the average over the inputs, multiplies it with a trainable scalar β , adds a trainable bias b , and passes the result through the non-linearity. The max pooling function

$$a_j = \max_{N \times N} (a_i^{n \times n} u(n, n)) \quad (2)$$

applies a window function $u(x, y)$ to the input patch, and computes the maximum in the neighborhood. In both cases, the result is a feature map of lower resolution.

3.4 Backpropagation

All layers of the CNN model are trained using the backpropagation algorithm. For error propagation and weight adaptation in fully connected, convolutional, and subsampling layers we follow the standard procedure. In the max pooling layers, the error signals are only propagated to the position at $\arg \max_{N \times N} (a_i^{n \times n} u(n, n))$. Thus, error maps in max pooling layers are sparse. If overlapping pooling windows are used, it might be necessary to accumulate several error signals in one unit.



Fig. 2. A few examples of preprocessed images from the Caltech-101 dataset



Fig. 3. Images from the NORB normalized-uniform dataset

4 Experimental Results

The purpose of our experiments is threefold: (1) to show that max pooling is superior to subsampling, (2) to determine if overlapping pooling windows can improve recognition performance and (3) to find suitable window functions. All experiments were conducted with the same random initialization unless noted otherwise.

To speed up the training, we implemented the CNN architecture on Graphics Processing Units (GPUs) using NVIDIA’s CUDA programming framework [16]. Convolution operations utilize routines kindly provided by Alex Krizhevsky¹. Most other operations are accelerated with our publicly-available CUV library [13]. For mini-batch learning, with only a few patterns being processed in parallel, we achieve a speedup of roughly two orders of magnitude compared to our CPU implementation.

4.1 Datasets

We evaluated different pooling operations on the Caltech-101 [4] and NORB [11] datasets. Recognition rates with other CNN architectures have been published for both datasets by various authors.

The Caltech-101 dataset consists of 101 object categories and one background category. There is a total of 9144 images of different sizes of roughly 300×300 pixels. We preprocessed the images by fitting them into a 140×140 image frame, while retaining their aspect ratio. The padding was filled with the image mean for each color channel. We faded the image border into the padding to remove side effects caused by the image edge, as shown in Figure 2. The resulting images are normalized per channel to have a mean of zero and a variance of one to accelerate learning.

We follow the common experiment protocol in the literature for Caltech-101, which is to randomly choose 30 images from each category for training and use

¹ <http://www.cs.utoronto.ca/~kriz>

the remainder for testing. The recognition rate is measured for each class and we are reporting the average over all 102 classes.

In addition, we conduct experiments on the NORB normalized-uniform dataset, which consists of only five object categories. The training and testing set each contain five instances of toy objects for each category, photographed from different angles and under different illuminations. Each pattern consists of a binocular pair of 96×96 grayscale images, with a total of 24,300 training patterns and the same amount of testing patterns.

4.2 Max Pooling versus Subsampling

In order to keep our results comparable, we deliberately designed the networks to resemble the one by Huang and LeCun [7] for NORB and the one by Ahmed *et al.* [1] for Caltech-101.

For NORB, the input layer with two feature maps of size 96×96 is followed by a convolutional layer C1 with 5×5 filters and 16 maps of size 92×92 . P2 is a 4×4 pooling layer, reducing the size to 23×23 . Convolutional layer C3 employs 6×6 filters and has 32 maps with dimensions of 18×18 pixels. Pooling layer P4 with 3×3 pooling windows yields 6×6 feature maps which are fully connected to 100 hidden neurons. The output layer takes input from all 100 neurons and encodes the object class with 5 neurons. This six-layer network is shown in Figure 11.

Two variations of the network were trained: In the first case, the pooling layers performed a subsampling operation, in the second case, this was replaced with max pooling. Note that the subsampling network has slightly more parameters owing to the trainable scalar and bias, as described in Section 3. After 1,000 epochs of backpropagation training with mini-batches of 60 patterns, there is a striking discrepancy between the recognition rates. For each network, five trials with different weight initializations were performed and we are reporting the average error rate as well as the standard deviation. The subsampling network achieves a test error rate of 7.32% ($\pm 1.27\%$), compared to 5.22% ($\pm 0.52\%$) for the max pooling network.

For Caltech-101, the input layer consists of three feature maps of size 140×140 , followed by a convolutional layer C1 with 16×16 filters and 16 feature maps. The subsequent pooling layer P2 reduces the 125×125 maps with 5×5 non-overlapping pooling windows to a size of 25×25 pixels. Convolutional layer C3 with 6×6 filters consists of 128 feature maps of size 20×20 . Pooling layer P4 uses a 5×5 non-overlapping pooling window. The neurons of those 128 feature maps of size 4×4 are fully connected to the 102 output units.

After 300 epochs using the RPROP [19] batch-learning algorithm, we evaluated the normalized recognition performance for Caltech-101. When using a subsampling operation, the network achieved a test error rate of 65.9%. Substituting this with a max pooling operation yielded an error rate of 55.6%.

For both NORB and Caltech-101 our results indicate that architectures with a max pooling operation converge considerably faster than those employing a subsampling operation. Furthermore, they seem to be superior in selecting invariant features and improve generalization.

Table 1. Recognition rates on NORB normalized uniform (after 300 epochs) and Caltech-101 (after 400 epochs) for networks with different amounts of overlap in the max pooling layers

	NORB		Caltech-101	
	train set	test set	train set	test set
no overlap	0.00%	6.40%	1.28%	52.29%
2 pixels overlap	0.00%	6.48%	2.29%	52.74%
4 pixels overlap	0.00%	6.37%	3.92%	52.42%
6 pixels overlap	0.01%	7.27%	4.55%	53.82%
8 pixels overlap	0.00%	6.84%	7.43%	55.79%
10 pixels overlap	0.01%	7.21%	10.17%	57.32%

Table 2. Test error rates for NORB (after 500 epochs of training) and for Caltech-101 (after 600 epochs). Applying a window function to an overlapping neighborhood is consistently worse than using non-overlapping pooling windows.

	No overlap	Rectangular	Cone	Pyramid	Triangle	Binomial
NORB test error	5.56%	5.83%	6.29%	6.28%	10.92%	12.15%
Caltech-101 test error	52.25%	57.77%	52.36%	51.95%	69.95%	73.16%

4.3 Overlapping Pooling Windows

To evaluate how the step size of overlapping pooling windows affects recognition rates, we essentially used the same architectures as in the previous section. Adjusting the step size does, however, change the size of the feature maps and with it the total number of trainable parameters, as well as the ratio between fully connected weights and shared weights. Therefore, we are increasing the size of the input feature maps accordingly, placing the input pattern in the center of a feature map and zero-padding it. In the NORB architecture, for example, the input feature maps are of size 106×106 , if a step size of two is chosen.

Table 1 lists the recognition rates for different step sizes on both datasets. The performance deteriorates if the step size is increased. This might be owed to the fact that there is no information gain if pooling windows overlap. Maxima in overlapping window regions are merely duplicated in the next layer and neighboring pixels are more correlated.

4.4 Window Functions

Small variations of the input image and shifts past the border of the pooling window can dramatically change the representation. For this reason we experimented with smoother, overlapping pooling windows. Window functions are

often used to smooth an input signal in signal processing applications. We have evaluated four different window functions, as shown in Table 2.

Again, the network architecture for those experiments was similar as in the previous sections. For the NORB dataset, the network was modified to receive 128×128 inputs and P2 pools from a 12×12 window with an overlap of 8 pixels. Units in P4 receive input from 9×9 windows, which are overlapping by 6 pixels. Thus, if a small rectangular window function is chosen, this is equivalent to the non-overlapping network. Similarly, for Caltech-101, the input was padded to 230×230 and layers P2 and P4 are pooling from 15×15 and 18×18 windows, respectively.

As shown in Table 2, none of the window functions improved recognition rates significantly. The binomial window function and the triangle window function even yield substantially worse results than a rectangular window function.

4.5 Results on Other Datasets

To our knowledge, no results using max pooling operations have been published on the MNIST dataset of handwritten digits [10] and the NORB jittered-cluttered dataset [11]. Therefore, we applied our network model with non-overlapping max pooling windows to both datasets.

For MNIST, we achieved 0.99% testing error with a rather shallow architecture. Here, the 28×28 input layer was followed by a convolutional layer C1 with 9×9 filters and 112 feature maps. In the subsequent max pooling layer P2 each unit receives input from a 5×5 window and is connected to each of the ten output neurons. We trained this architecture for 60 epochs of online updates with backpropagation.

The NORB jittered-cluttered dataset was evaluated with the following architecture: The stereo input patterns of size 108×108 are convolved in C1 with 5×5 filters into 16 feature maps and max pooled in P2 with 9×9 windows and an overlap of 4 pixels. They are convolved with 8×8 filters in C3 to produce 32 feature maps of size 13×13 . Layer P4 reduces the size to 5×5 pixels by pooling from 5×5 windows with an overlap of 3 pixels. Two fully connected layers with 100 neurons and six output neurons conclude the architecture. Training this model for seven epochs with online learning and mini-batches of 60 patterns yielded a test error rate of 5.6%. To our knowledge, this is the best published result on this dataset so far.

We also improved our results on NORB normalized-uniform with a few refinement passes using an emphasizing scheme. During these passes, difficult patterns with above-average errors were trained more often. Here, we achieved a test error rate of 4.57%, which exceeds the 5.2% reported by Nair and Hinton [15], even though they used additional unlabeled data.

5 Conclusion

We have shown that a max pooling operation is vastly superior for capturing invariances in image-like data, compared to a subsampling operation. For several

datasets, recognition results with an otherwise equivalent architecture greatly improve over subsampling operations. On NORB normalized-uniform (4.57%) and NORB jittered-cluttered (5.6%) we even achieved the best results published to date.

However, using smoother, overlapping pooling windows does not improve recognition rates. In future work, we will investigate further into finding suitable window functions. Histograms (as in [312]) can be seen as a third kind of pooling operation which has not yet been thoroughly evaluated. Combining such histogram operations with Convolutional Neural Networks might further improve recogniton rates on vision tasks.

References

1. Ahmed, A., Yu, K., Xu, W., Gong, Y., Xing, E.: Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 69–82. Springer, Heidelberg (2008)
2. Behnke, S.: Hierarchical Neural Networks for Image Interpretation. LNCS, vol. 2766. Springer, Heidelberg (2003)
3. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: CVPR, pp. 886–893 (2005)
4. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding 106(1), 59–70 (2007)
5. Frome, A., Cheung, G., Abdulkader, A., Zennaro, M., Wu, B., Bissacco, A., Adam, H., Neven, H., Vincent, L.: Large-scale Privacy Protection in Google Street View. EUA, California (2009)
6. Fukushima, K.: A neural network model for selective attention in visual pattern recognition. Biological Cybernetics 55(1), 5–15 (1986)
7. Huang, F.-J., LeCun, Y.: Large-scale learning with svm and convolutional nets for generic object categorization. In: Proc. Computer Vision and Pattern Recognition Conference (CVPR 2006). IEEE Press, Los Alamitos (2006)
8. Hubel, D.H., Wiesel, T.N.: Receptive fields of single neurones in the cat's striate cortex. The Journal of Physiology 148(3), 574 (1959)
9. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: CVPR, vol. (2), pp. 2169–2178. IEEE Computer Society, Los Alamitos (2006)
10. LeCun, Y., Bottou, L., Orr, G., Müller, K.: Efficient BackProp. In: Orr, G.B., Müller, K.-R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, p. 9. Springer, Heidelberg (1998)
11. LeCun, Y., Huang, F., Bottou, L.: Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In: Proceedings of CVPR 2004. IEEE Press, Los Alamitos (2004)
12. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60, 91–110 (2004)
13. Müller, A., Schulz, H., Behnke, S.: Topological Features in Locally Connected RBMs. In: Proc. International Joint Conference on Neural Networks, IJCNN 2010 (2010)

14. Mutch, J., Lowe, D.G.: Multiclass Object Recognition with Sparse, Localized Features. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 11–18 (2006)
15. Nair, V., Hinton, G.: 3-d object recognition with deep belief nets. In: Advances in Neural Information Processing Systems (2010)
16. Nvidia Corporation. CUDA Programming Guide 3.0 (February 2010)
17. Osadchy, M., LeCun, Y., Miller, M.: Synergistic Face Detection and Pose Estimation with Energy-Based Models. *Journal of Machine Learning Research* 8, 1197–1215 (2007)
18. Ranzato, M., Huang, F.-J., Boureau, Y.-L., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: Proc. Computer Vision and Pattern Recognition Conference (CVPR 2007). IEEE Press, Los Alamitos (2007)
19. Riedmiller, M., Braun, H.: RPROP – Description and Implementation Details. Technical report, University of Karlsruhe (January 1994)
20. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. *Nature Neuroscience* 2, 1019–1025 (1999)
21. Serre, T., Wolf, L., Poggio, T.: Object recognition with features inspired by visual cortex. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 2 (2005)
22. Siagian, C., Itti, L.: Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(2), 300 (2007)
23. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best Practice for Convolutional Neural Networks Applied to Visual Document Analysis. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 958–962. IEEE Computer Society, Los Alamitos (2003)

Visual Shape Recognition Neural Network Using BESOM Model

Hiroaki Hasegawa and Masafumi Hagiwara

Department of Information and Computer Science, Keio University,
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522 Japan
{hasegawa,hagiwara}@soft.ics.keio.ac.jp

Abstract. In this paper, we propose a neural network recognizing visual shapes based on the Bidirectional SOM (BESOM) model. The proposed network has 4 features. First, the network is based on the BESOM model, which is a computational model of the cerebral cortex. Second, the Gabor filter, a model of a simple cell in the primary visual area, is used to calculate input features. Third, the network structure mimics the ventral visual pathway of the brain, which is said to recognize visual shapes. Finally, this is the first application of the BESOM model which is large-scale and multi-layer as far as we know. We conducted an experiment to assess the network and confirmed that it can recognize alphabets.

Keywords: BESOM, Neural Network, Gabor Filter, Ventral Visual Pathway, Object Recognition.

1 Introduction

The vision is one of the most important abilities for humans to get information about the real world. The same can be said for human-like robots – it is a necessary technique to process the camera image and extract some information.

Human-brain functions are sometimes modeled and implemented using neural networks. Fukushima proposed the neocognitron [1] imitating the ventral visual pathway. Since it is a robust classifier, many applications have been reported [2,3,4]. Cortex [5] is also a model of the ventral visual pathway.

The BESOM [6,7,8] is another computational model of the cerebral cortex. Its network is a kind of Bayesian network [9] whose nodes are replaced with Self-Organizing Maps (SOMs) [10]. Because a circuit to realize BESOM network is highly similar to anatomical property in the cerebral cortex [7], it can be considered as one of plausible models of it. There are theoretical studies or small-scale experiments [8,11], however, no large-scale or multi-layer application based on this model has been reported as far as we know.

In this paper, we propose a neural network recognizing visual shapes based on the BESOM model and apply it to an alphabet recognition task. Input features are computed using the Gabor filter [12], which is considered to be a model of a simple cell in the primary visual area [13]. The features are fed to the input layer

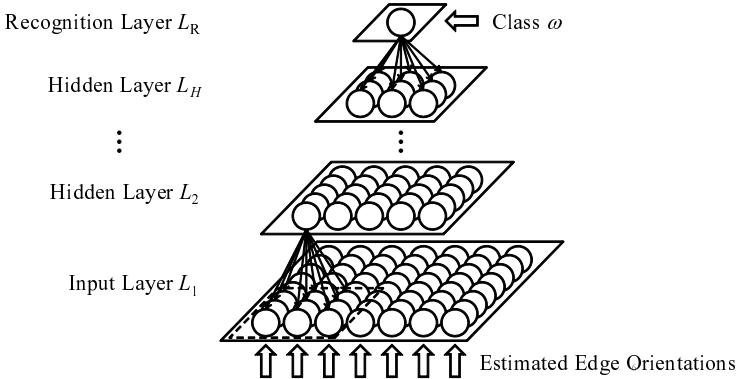


Fig. 1. Network structure

of the network and then travel through hidden layers to the last layer. The firing pattern in the last layer eventually determines the class of the input image.

This paper is organized as follows. In Sect. 2 we detail our proposed network. Section 3 shows the result of an experiment in order to evaluate the proposed network. We conclude the paper in Sect. 4.

2 Visual Shape Recognition Neural Network Using BESOM Model

2.1 Object Recognition

An object recognition task proceeds as follows. First, a gray-scale input image is given to the network. Next, the image is preprocessed and the result is used as an observed data. Then the recognition step is carried out to estimate the Most Probable Explanation (MPE) and thus the recognition result becomes available.

Preprocessing. At first the network apply Gabor filters to the input image. When applied, an orientation is quantized in a few levels.

Applying Gabor filters, representative edge orientations are estimated locally in each subarea, called *receptive field*. They are fed to the input layer of the network as observed data.

Network Structure. Figure 1 depicts the overview of the proposed network. At the bottom is placed the input layer L_1 and it is connected indirectly through hidden layers L_2, L_3, \dots, L_H to the recognition layer L_R . This mimics a model of the ventral visual pathway in the cerebral cortex [14][15].

Nodes in L_1 are torus 1-dimensional probability distribution SOMs [8]. Their units represent edge orientations. When observed data is given, a unit corresponding to the estimated edge orientation is selected.



Fig. 2. Sample images from the training and test datasets. The left 3 images are the letters ‘W’ of Barret, Meiryo, and Times New Roman in the training dataset. The rest 3 are of Antique 101, Candara, and Minion Pro in the test one.

Nodes in two adjacent layers are connected; in terms of Bayesian network, they makes causal relationships. Because they are also 1-dimensional probability distribution SOMs, geometrical information is passed up in some compressed representation.

There is only one node in the recognition layer L_R . Each unit in the node represents an object class ω , one by one.

Recognition Step. In the recognition step, the network estimates the MPE of the observed data. At this time, only bottom-up Bayesian inference is performed. A belief revision algorithm might be suitable if it is implementable.

A recognition result (class ω) can be obtained from the recognition layer L_R .

2.2 Learning

Learning is done by two phases: the self-organizing and the class learning ones.

In the self-organizing phase no supervisor signal is given. Conditional probability tables (connection weights) among lower layers are learned.

Class learning phase gives the network a supervised signal ω_t to L_R . Weights are learned for mapping input images to their own classes.

3 Experiment

We carried out an experiment to evaluate the proposed network. The task is to classify synthetic alphabetical images into respective classes. We construct 6-layer network and use 1,040 images in total. Figure 2 exhibits some of the images for example.

The network recognized alphabets in the training dataset with the precision score of 88.5 %, whereas those in the test one with 87.7 %. As far as we check misclassifications, most faults occurred because of presence/absence of edges. This can be resolved by introducing the belief revision algorithm for MPE estimation.

4 Conclusion

In this paper, we presented our proposed neural network which recognizes visual shapes. It is based on the BESOM model and is the first application which is large-scale and multi-layer, as far as we know.

We applied this network to an alphabet recognition task. The result shows that the proposed network can recognize most alphabet images. Further studies for improving performance will be the focus of our future work.

Acknowledgments. The authors would like to thank Y. Ichisugi for his preceding studies and all the advice and help he gave us.

References

1. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* 36(4), 193–202 (1980)
2. Fukushima, K.: Neocognitron for handwritten digit recognition. *Neurocomputing* 51, 161–180 (2003)
3. Terajima, T., Hagiwara, M.: Image recognition system with parallel-hierarchical neural network based on visual information processing. *IEICE Technical Report. NC.* 107(542), 243–248 (2008) (in Japanese)
4. Shimomura, M., Sato, S., Miyake, S., Aso, H.: New neocognitron-type network and its learning method based on ICA and PCA (in Japanese). *IEICE Transactions J88-D-II(4)*, 769–777 (2005)
5. Schrader, S., Gewaltig, M.O., Körner, U., Körner, E.: Context: A columnar model of bottom-up and top-down processing in the neocortex. *Neural Networks* 22, 1055–1070 (2009)
6. Ichisugi, Y.: A cerebral cortex model that self-organizes conditional probability tables and executes belief propagation. In: *IJCNN 2007*, pp. 178–183 (2007)
7. Ichisugi, Y.: Elucidation progress of the information processing principle of the brain. *Technical Report AIST07-J00012*, AIST (2008) (in Japanese)
8. Ichisugi, Y.: BESOM Ver.1.0: Algorithm of the cerebral cortex. *Technical Report AIST09-J00006*, AIST (2009) (in Japanese)
9. Bishop, C.: *Pattern Recognition and Machine Learning.*, pp. 360–372. Springer Science+Business Media, New York (2006)
10. Kohonen, T.: *Self-Organizing Maps (Revised Edition)*. Springer, Japan (2005) (in Japanese)
11. Hosoya, H.: Learning and recognition of hierarchical temporal series based on BE-SOM model (in Japanese). *IEICE Technical Report. NC* 107, 151–156 (2008)
12. Jones, J., Palmer, L.: An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology* 58(6), 1233–1258 (1987)
13. Hubel, D., Wiesel, T.: Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology* 195, 215–243 (1968)
14. Felleman, D., Van Essen, D.: Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex* 1, 1–47 (1991)
15. Tanaka, H., Fujita, I.: Perceiving and recognizing the shape of an object. In: *Knowing the Brain*, pp. 74–85. Shujun-sha, Tokyo (1999) (in Japanese)

Comparing Feature Extraction Techniques and Classifiers in the Handwritten Letters Classification Problem

Antonio García-Manso, Carlos J. García-Orellana,
Horacio M. González-Velasco, Miguel Macías-Macías,
and Ramón Gallardo-Caballero

CAPI Research Group, Elvas Av., 06006 Badajoz, Spain
antonio@capi.unex.es

Abstract. The aim of this study is to compare the performance of two feature extraction techniques, Independent Component Analysis (ICA) and Principal Component Analysis (PCA) and also to compare two different kinds of classifiers, Neural Networks and Support Vector Machine (SVM). To this aim, a system for handwritten letters recognition was developed, which consist of two stages: a feature extraction stage using either ICA or PCA, and a classifier based on neural networks or SVM. To test the performance of the system, the subset of uppercase letters of the NIST#19 database was used. From the results of our tests, it can be concluded that when a neural network is used as classifier, the results are very similar with the two feature extraction techniques (ICA and PCA). But when the SVM classifier is used, the results are quite different, performing better the feature extractor based on ICA.

Keywords: Principal component analysis, independent component analysis, neural networks, support vector machines, handwritten letters recognition.

1 Introduction

Handwritten letters classification is a typical example of complex pattern recognition system. In this problem, it is difficult to achieve high rates of accuracy using only a set of features and a single classifier, since handwritten letters contain many pattern variations which mostly depend upon individual writing styles. As mentioned above, it is not intended in this work to develop a robust system for classifying handwritten uppercase letters, but only to compare the performance obtained with two techniques of features extraction, ICA [1] and PCA [2] and also two different kinds of classifiers, neural networks [3] and SVM [4]. To this end, the classification of handwritten letters was chosen because it is a very complex problem, and clearly allows us to explore the performance of each of the above-mentioned techniques.

This paper is organized as follows. Section II presents a description of the methodology used to carry out the work. Section III compares the results

obtained by the two feature extraction techniques. Finally in Section IV the main conclusions of this work are presented.

2 Outline of the Process

The process devised for pattern classification is shown in Figure 1.

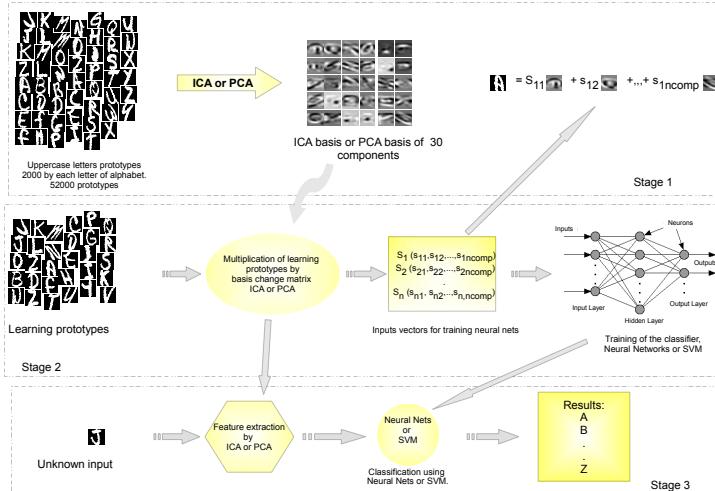


Fig. 1. Outline of the system's operation

As it can be seen in that figure the training process has three stages.

- **Stage 1:** In this stage, ICA or PCA basis are obtained using the prototypes selected randomly from NIST#19 database [5]. These basis are generated by applying the *fastICA* or the *prcomp* functions (developed with the *R* language [6]) respectively. In our tests, we took as classes each of the 26 letters of the Western Alphabet, and selected randomly 52,000 prototypes of the 26 classes (2,000 of each class, with a dimension of 32×32 pixels for each one) in order to obtain the matrix for the change of basis.
- **Stage 2:** In this second stage, we obtained the new vectors of prototypes, with a lower number of components, by the multiplication of the change of basis change (obtained in the stage 1) and the prototypes of the classes. These new vectors of prototypes (learning vectors) were divided in three subsets (training, 800 of each class, validation and test, 200 of each class) [7]. Once the files of prototypes for learning, validation and test were generated, the neural network (*Multilayer Perceptron*) and the SVM classifiers were trained.

- **Stage 3:** Finally, in the last step, unknown inputs vectors, selected from hsf_4 [8], were classified with the neural networks and the SVM classifiers trained in the stage 2. To this end, a feature extraction was carried out over the unknown input vectors using the change of basis matrix generated in stage 1. This was made to check whether the system was able to generalize, that is, the system’s performance was being tested at this stage.

3 Results

As commented in Section 2, to compare the performance of ICA and PCA as feature extractors, and also to compare the performance of the multilayer perceptron and SVM as classifiers, our method was applied to the uppercase letters contained in the NIST#19 database. For the test subset, prototypes from hsf_4 partition were selected randomly. In Figure 2, the best results are shown for the neural network classifier and for the SVM classifier respectively, and in both cases the results are presented using ICA and PCA as feature extractor.

Regarding the ICA-PCA comparison as feature extractors, it can be seen (in Figure 2) that the result are almost identical when using the neural networks as classifier, but significantly different with SVM. In this last case, for a low (10-15) or high (60-65) number of components in the input vectors the results are similar, but however, for a number of components in the range of 20-55, ICA performs significantly better than PCA.

On the other hand, the best performance of the neural network classifier and the SVM classifier are very similar when the feature extraction is made with

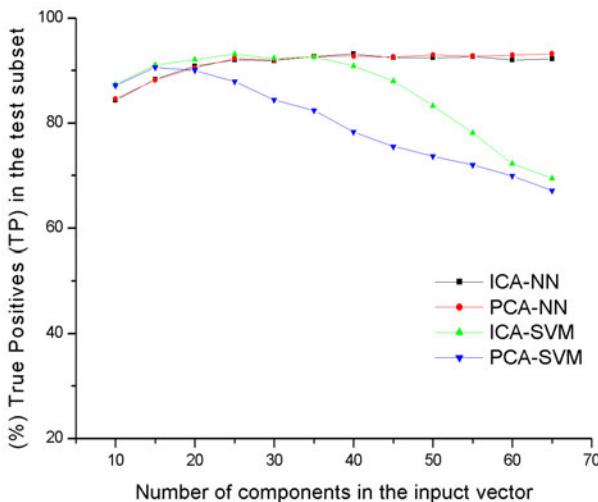


Fig. 2. ICA-PCA and NN-SVM comparisons, using the same learning, validation and test sets

ICA, but is better for the neural network classifier when the feature extraction technique used is PCA. It can be observed in Figure 2 that, with PCA, the performance of both classifiers is very similar with low dimensions of the input vectors (up to 35 components), but when this dimension raises, the performance of SVM decreases a lot, while the performance of the neural network remains the same.

Finally, it is important to highlight that the best results obtained with our method, whatever the feature extractor of the classifier, are quite good, comparable with any of the other method proposed.

4 Conclusions

As mentioned in the introduction, the aim of this work was not to develop a classification system for handwritten characters to outperform the existing ones (even commercials), but to compare two feature extraction techniques, ICA and PCA, and two method of classification, Neural Networks and SVM classifiers. However, with the proposed system, apart from comparing the mentioned techniques, a system was achieved that can be put side, in performance, to those we have found in the literature for handwritten letters classification. From results of our tests, it can be concluded that when a neural network is used as classifier, the results are very similar with the two feature extraction techniques (ICA and PCA). But when the SVM classifier is used, the results are quite different, performing better the feature extractor based on ICA. Finally, both classifiers perform identically when the most suitable feature extractor is used.

Acknowledgments

This work has been partly supported by “Junta de Extremadura” and FEDER through projects PRI08A092 and PDT09A036.

References

1. Hyv  inen, A., Oja, E.: Independent components analysis:algorithms and applications. *Neural Networks* 13, 411–430 (2000)
2. Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Series in Statistics. Springer, Heidelberg (2002)
3. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, Englewood Cliffs (1999)
4. Vapnik, V.N.: The Nature of Statistical Learning Theory, 2nd edn. Statistics for Engineering and Information Science. Springer, Heidelberg (2000)
5. Grother, P.J.: Nist special database 19: handprinted forms and characters database. Technical Report and CDROM (1995)
6. The r project for statistical computing, <http://www.r-project.org/>
7. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
8. D'Amato, D.P. (ed.): Cross Validation Comparison of NIST OCR Databases. SPIE, Bellingham (April 1993)

The Parameter Optimization of the Pulse Coupled Neural Network for the Pattern Recognition

Masato Yonekawa and Hiroaki Kurokawa

School of Computer Science, Tokyo University of Technology,
1404-1, Kataura, Hachioji, Tokyo, Japan

Abstract. The pattern recognition using Pulse Coupled Neural Network (PCNN) had been proposed. In conventional studies, the parameters in the PCNN are used to be defined empirically and the optimization of parameters has been known as a remaining problem of PCNN. In this study, we show a method to apply the real coded genetic algorithm to the parameter optimization of the PCNN and we also show performances of pattern recognition by the PCNN with learned parameters.

Keywords: PCNN, pattern recognition, real coded genetic algorithm.

1 Introduction

The PCNN had been proposed as a numerical model of the cat visual cortex[1] and many applications to the image processing had been proposed recently. Recognizing objects is an important issue in image processing and the method using PCNN had also been proposed in conventional studies[2][3][4]. The PCNN consists of a lot of neurons which has two kinds of inputs (feeding input and linking input) with leaky integrators. For the image processing, PCNN has two-dimensional structure with lattice connection among neurons and each neuron corresponds to the pixel of the image. The pattern recognition method using PCNN is based on the PCNN's synchronous dynamics. A number of firing neurons in the PCNN in every time step shows a temporal pattern corresponds to the input pattern. This firing temporal pattern is defined as a *PCNN icon* and this shows unique characteristics for every pattern to be recognized[2]. The PCNN has a lot of parameters to be defined and its dynamics depends on the parameters. In conventional studies, the parameters are assumed to be appropriately defined, that is, the parameters are practically defined by trial and error.

In this study, we implement the real coded genetic algorithm to the PCNN parameter optimization and show the validity of the method for the pattern recognition using PCNN. An efficiency of our proposed method will be shown in the simulation results.

2 The Parameter Optimization Method Using Real Coded GA

The pattern recognition using PCNN is achieved by means of the PCNN icon which is proposed in the conventional study[2]. Here the PCNN icon is defined as the time series of the number of firing neurons in the PCNN. The PCNN icon is unique to the input pattern and its form is almost independent of the rotation and magnification or shrinking of

the pattern. To obtain the PCNN icons, a number of firing neurons are observed in every time step from $t = 0$ to $t = t_{\max}$, where t_{\max} is defined arbitrary. Namely, the PCNN icon is also defined as a t_{\max} -dimensional real number vector. The pattern recognition using PCNN is based on the similarity of these PCNN icons. Since the dynamics of the PCNN depends on its parameters, the parameters optimization technique has been required to avoid a false recognition. However, in conventional studies, the parameters in the PCNN are properly defined to obtain the successful results.

In this study, the real coded genetic algorithm is applied to the optimization of the parameters in PCNN for the pattern recognition. We assumed that the parameters to be optimized are β_{ij} , τ_L , τ_T , V_L , and V_T . Here, the definitions and details of these parameters are shown in our previous study [5] or other conventional studies. To apply the real coded genetic algorithm to the parameter optimization, a set of real numbers of these 5 parameters are defined as a chromosome. Here, the ranges of the real numbers are assumed to be $0 \sim 20.48$ and 17 chromosomes are generated randomly as an initial population.

The fitness is calculated based on the normalized correlation coefficient among the PCNN icons corresponding to the learning images. In proposed method, we assumed that plural images per pattern are prepared as learning images. Here the normalized correlation coefficient between the PCNN icons of image X and Y is given by,

$$\text{correlation} = \frac{\sum_{t=0}^{t_{\max}} (x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum_{t=0}^{t_{\max}} (x_t - \bar{x})^2} \sqrt{\sum_{t=0}^{t_{\max}} (y_t - \bar{y})^2}}. \quad (1)$$

Where, x_t and y_t are the number of firing neurons at time step t , and \bar{x} and \bar{y} are the average of x_t and y_t through $t = 0$ to $t = t_{\max}$.

The fitness is calculated as shown in following procedure. Where we assumed that the patterns to be learned are P_1, P_2, \dots, P_n , and each pattern has m ($m \geq 2$) images as a learning image. Note that a rotated angle or a reduced scale of these m images is assumed to be different each other.

1. Calculate PCNN icons for all learning images using parameters in j th chromosome.
2. Calculate the correlation coefficient among PCNN icons of all learning images.
3. Search the minimum correlation coefficient among the same pattern learning images for pattern P_i and substitute it for a_i .
4. Search the maximum correlation coefficient among the different pattern learning images for pattern P_i and substitute it for b_i .
5. Repeat 2 and 3 for every P_i ($i = 1, 2, \dots, n$)
6. Calculate the difference between the minimum a_i and the maximum b_i and substitute it for C_j , where j is the chromosome number.
7. Calculate C_j 's for all chromosomes (repeat 1-6) and sort it in ascending order.
8. Use the order which is obtained in step 7 as a fitness of the chromosome.

To breed a new generation, two chromosomes are selected depending on the fitness using roulette-wheel selection and the one-point crossover is applied to the selected chromosome. The mutation is also applied to the selected chromosomes. In the mutation, one randomly selected parameter in the chromosome will be changed to random value. Here, the probability of the crossover and mutation are 0.6 and 0.2, respectively. These

procedure will continue until 16 chromosomes are produced. Also, in our algorithm, elitist strategy is used. The chromosome which has largest fitness is selected and carry over to the next generation. Then 17 chromosomes will be produced as a population for the next generation. We assumed that the procedure of breeding the next generation is terminated in 500th generation.

3 Simulation Results

Simulation results of pattern recognition using the PCNN which is applied our parameter optimization method are shown in this section. The learning images are shown in Fig. 1. Where we assumed that the Fig. 1(a) and (c) show the original pattern images and Fig. 1(b) and (d) show the modified pattern images. The results of the parameter learning using our algorithm are shown in Table 1. Desirable correlation among original and modified pattern is obtained as a result of the parameter optimization.

Also, we show the simulation results using test images. The test images are modified in scale or angle from the original images which are shown in Fig. 1(a) and (c). The correlation coefficients between each of the original images and the test images are summarized in Table 2. From the results, all the test images are perfectly recognized as a correct pattern. Here, we assumed that the 0.5 is the threshold of the correlation coefficient to consider as the same pattern. In the results, the correlation coefficients between 70% shrinking image of "T" and its original image was calculated comparatively small value and it is considered that the pattern recognition using PCNN has a possibility to lead a false recognition. However, this problem is considered as a matter of an image resolution, that is, only 11×11 pixels are used to display the pattern.

Figure 2(a) and (b) show summaries of the correlation coefficients between the original pattern images of "+" and "T" and test images whose angle and scale are modified from the original patterns. In the figures, each vertical and horizontal axis corresponds to a rotated angle and a shrinking scale, respectively, and the correlation coefficient between the original pattern image and the modified image is indicated by grayscale. From

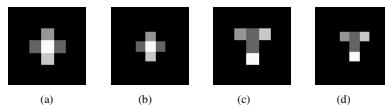


Fig. 1. The learning images used for the parameter optimization, (a) learning image of the pattern "+" (original image), (b) 20% shrinking image of (a), (c) Learning image of the pattern "T" (original image), (d) 20% shrinking image of (c). The size of all images is 66×66 pixels.

Table 1. A summary of the correlation coefficients among PCNN icons of learning patterns

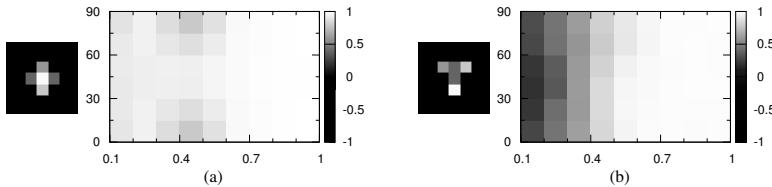
	(a)	(b)	(c)	(d)		(a)	(b)	(c)	(d)
original "T" (a)	1	0.818	0.790	0.805	original "T" (a)	1	0.988	0.008	0.020
shrinking "T" (b)		1	0.764	0.664	shrinking "T" (b)		1	0.000	0.013
original "+" (c)			1	0.954	original "+" (c)			1	0.995
shrinking "+" (d)				1	shrinking "+" (d)				1

(a) The correlation coefficients before learnig

(b) The correlation coefficients after learnig

Table 2. The correlation coefficients between the original images and the test images

modified "+"						modified "T"					
angle			scale			angle			scale		
30	60	90	0.7	0.5	0.3	30	60	90	0.7	0.5	0.3
original "+"	0.997	0.997	1.000	0.976	0.776	0.952	0.002	0.002	0.008	0.005	0.006
original "T"	0.010	0.012	0.008	0.001	0.117	-0.005	0.994	0.982	1.000	0.982	0.890
											0.506

**Fig. 2.** A summary of the correlation coefficient for rotation and enlargement or reduction

the results in Fig. 2, we can know that the rotated pattern images are perfectly recognized in any angle. On the other hand, the correlation coefficient between the original images and the scale-modified images decreased in highly shrinking scale due to the matter of an image resolution, as we mentioned above. However, almost of the results show that the correlation coefficients between the original pattern and the modified pattern is comparatively large and the recognition of the test pattern is feasible in the case that the threshold is set to 0.5.

4 Conclusion

In this study, we described a method to apply the genetic algorithm to the optimization of the parameters in PCNN for image recognition. The real coded genetic algorithm was used for the optimization and the fitness is calculated based on the correlation coefficients among PCNN icons. Simulation results showed good performances of the PCNN with our algorithm for the pattern recognition.

Acknowledgement. This work was supported by KAKENHI (No. 22650046).

References

1. Echorn, R., et al.: Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. *Neural Computation* 2, 293–307 (1990)
2. Johnson, J.L., Padgett, M.L.: PCNN Models and Applications. *IEEE Trans. Neural Network* 10(3), 480–498 (1999)
3. Gu, X., et al.: Object Detection Using Unit-Linking PCNN Image Icons. In: Wang, J., Yi, Z., Žurada, J.M., Lu, B.-L., Yin, H. (eds.) *ISNN 2006. LNCS*, vol. 3971, pp. 616–622. Springer, Heidelberg (2006)
4. Mahgoub, A.G., et al.: An Intersecting Cortical Model Based Framework for Human Face Recognition. *J. of Systemics, Cybernetics and Informatics* 6(2), 88–93 (2008)
5. Yonekawa, M., Kurokawa, H.: An automatic parameter adjustment method of Pulse Coupled Neural Network for image segmentation. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) *ICANN 2009. LNCS*, vol. 5768, pp. 399–400. Springer, Heidelberg (2009)

The Use of Feed Forward Neural Network for Recognizing Characters of Dactyl Alphabet

Roman Zálusky, Emil Raschman, Mário Krajmer, and Daniela Ďuračková

Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology, Slovakia
`roman.zalusky@stuba.sk`

Abstract. This paper deals with recognition in dependency of count of objects in the training data set. For the capture the fingers position is used the sensory glove 5DT Data Glove Ultra. We describe the computer software "Dactyl Teacher" for recognition characters of Dactyl alphabet. This software contains a feed forward neural network, which is used for character recognition. In experimental part we examined count of iterations to train the neural network in influence of count of hidden layers and of count of neurons in these hidden layers. We choose the sufficient combination of these parameters for our feed forward neural network. Next we examined the success of recognition in dependence of count of training objects in training data set.

1 Introduction

The level of the performance of recognition is depending of number of objects in the training data set, network topology and learning algorithm. The finger talking is just one facility for deaf kids of the preschool age in today's conditions, with which it is possible to communicate without get the speech. The Dactyl alphabet is used by deaf children in many countries. This program Dactyl teacher will be helpful for learning the Dactyl alphabet using the computer.

2 Main Results

2.1 Dactyl Teacher and Sensory Glove

We developed the software Dactyl teacher for recognizing and learning the Dactyl alphabet. Character capture is realized with the use of a commercial sensory glove - 5DT Data Glove Ultra. On this glove there are 14 optical sensors. Out of each sensor is in range from 0 to 4096. This range is redundant, therefore we adjust it to range from 0 to 127. Between each pair of fingers there is a sensor which scans the diversion of these fingers. Each finger has two sensors which scan bending of fingers.

The captured data of finger positions from the sensory glove are next processed in our software Dactyl teacher. The software contains a feed forward neural network which ensures character recognition. Output of this neural network is 6-bits character code.

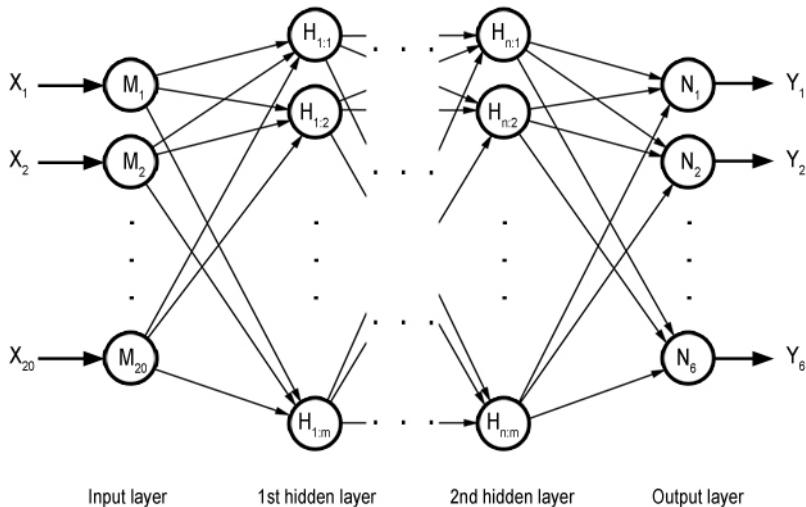


Fig. 1. The topology of feed forward neural network which is used in Dactyl teacher software

2.2 Neural Network

The neural network is a component of the Dactyl teacher software. This neural network consists of input layer, output layer and various count of hidden layers. Input layer contains 20 input neurons, output layer contains 6 neurons and count of neurons in hidden layers are also varied. The topology of the feed forward neural network is shown in Fig. 1. Each neuron is connected with each other neuron from previous layer. Output of this neural network is the 6-bits binary character code, which represents the recognized character. The output activity of neuron is expressed in equation (1) [1]. We have used as the activation function the sigmoidal function. We used the back propagation algorithm to train our feed forward neural network [2].

$$y = s\left(\sum_{i=1}^n w_i x_i + \vartheta\right) \quad (1)$$

where w_i and ϑ are the weight and threshold coefficients of the neuron, x_i are the activities of parent neurons and $s(x)$ is the activation function of the neuron.

2.3 Experimental

We examined count of iterations to training the neural network in influence of count of hidden layers and of count of neurons in these hidden layers. Count of hidden layers was changing in range from 1 to 5 layers and count of neurons in

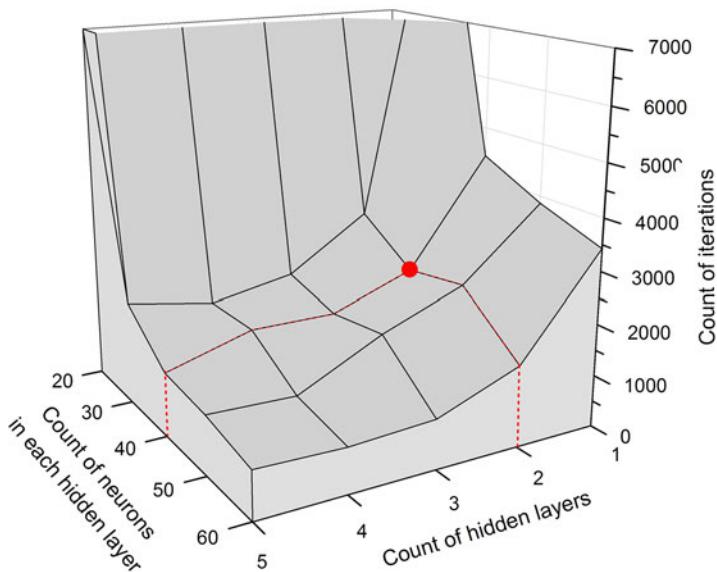


Fig. 2. Count of iterations in training the neural network in influence of count of hidden layers and of count of neurons in these hidden layers

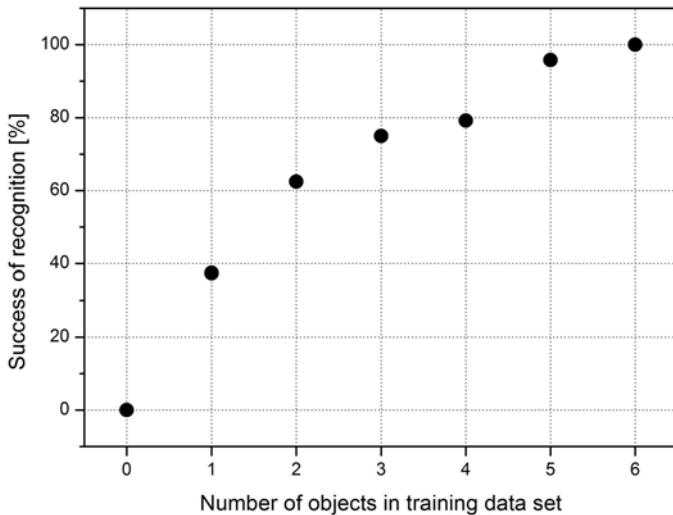


Fig. 3. Success of recognition in dependency of count of objects in training data set

each hidden layer was changing in range from 20 to 60 with step of 10 neurons. Count of neurons was the same for all hidden layers. We did 5 training processes for each combination and then we did the average value. The training processes were executed with learning coefficient λ equal to 0,01 and momentum parameter μ equal to 0,5. Initial weights and threshold coefficients were random generated from interval $<-1, 1>$. The obtained dependence is showed in Fig.2

From this dependence we chose neural network with two hidden layers and with 40 neurons in each hidden layer, because this neural network is relative simply and is able to solve our problem. Then to train this network is needed adequately count of iterations. Next we examined the success of recognition in dependency of count of objects in training data set for this neural network. To learn the neural network, we have used the training data set, which contain training objects in range from 1 to 6 for each character. Obtained dependence is showed on Fig.3. For full recognition must be the neural network learned to training data set, which contain minimal six training objects.

3 Conclusion

We have developed software Dactyl teacher, which is used for recognizing characters of Dactyl alphabet with use the sensory glove. The software contains also the feed forward neural network which ensures character recognition. First we examined the count of iterations to train the neural network in influence of count of hidden layers and count of neurons in each hidden layer. From this dependence we chose the neural network, which contains 2 hidden layers with 40 neurons in each hidden layer. Next we investigated success of recognition in dependency of count of objects in training data set for this neural network. For full recognition the neural network have to learned to training data set, which contain minimal six training objects.

Acknowledgement

This contribution was supported by the Ministry of Education Slovak Republic under grant VEGA No 1/0693/08.

References

1. Kvasnička, V., Benušková, Ľ., Farkaš, I., Kráľ, A., Pospíchal, P., Tiňo, P.: Introduction into the neural networks. IRIS, Bratislava (1997)
2. Alsmadi, M.K., Omar, K.B., Noah, S.A., Almarashdah, I.: Performance Comparison of Multi-layer Perceptron (Back Propagation, Delta Rule and Perceptron) algorithms in Neural Networks. In: IEEE International on Advance Computing Conference, IACC 2009, pp. 296–299 (2009)

Detecting DDoS Attack towards DNS Server Using a Neural Network Classifier^{*}

Jun Wu, Xin Wang, Xiaodong Lee, and Baoping Yan

China Network Information Center,

Computer Network Information Center, Chinese Academy of Sciences, Beijing, China

Graduate University of Chinese Academy of Sciences, Beijing, China

{wujun,wangxin,lee}@cnnic.cn, ybp@cnic.cn

Abstract. With the rapid growth of the distributed denial of service (DDoS) attacks, the identification and detection of attacks has become a crucial issue. In this paper we present a neural network approach to detecting the DDoS attacks towards the domain name system. A multi-layer feed-forward neural network is employed as a classifier based on the selected features that reflect the characteristics of DDoS attacks. The performance and the computational efficiency of the neural network classifier are both evaluated.

Keywords: DDoS, Domain name system, Feature extraction, Neural network, Classifier.

1 Introduction

Nowadays, distributed denial of service (DDoS) attacks are increasingly easy to implement. Network service providers have been frequently plagued by DDoS attacks. One of the services that has also been attacked is the Domain Name System (DNS) service.

DNS is one of the most important Internet infrastructure. Unfortunately, in recent years, DNS has become a primary prey of hackers. In the attacks towards DNS, DDoS attacks harm the most and the consequences are often disastrous. Therefore, the detection of DDoS attacks is vital for the smooth operation of DNS. There are two prevalent DDoS attacks against DNS servers: UDP flooding and scripting attack. In this paper we present a method of detecting the scripting attack.

DDoS detection methods are extensively studied by the research communities. Complex processing techniques like clustering algorithms [1] are promising but not readily available since they involve high processing overhead. Moreover, most DDoS detection approaches are based on single feature, such as entropy of the source IP address, which is insufficient for detecting complex attacks. In [2],

* This research is supported by the China Next Generation Internet Project: Industrialization of Next Generation Trusted Domain Name Service System (Grant number: CNGI-09-03-04).

an intrusion detection system for DoS attacks against DNS using neural network is introduced. However, it is working at traffic level and extracts 3 statistical features to characterize the DNS traffic, which does not involve the packet content to summarize the DNS query pattern.

Neural networks are the most commonly used soft computing technique in Intrusion Detection Systems (IDSs). *Ryan et al.* [3] described an off-line anomaly detection system which utilized a back-propagation multi-layer perceptron (MLP) neural network. *Cunningham and Lippmann* [4] used neural networks in misuse detection to detect Unix-host attacks by searching for attack specific keywords in the network traffic.

We begin in section 2 with the description of the features selected as inputs to the neural network and the configuration of the classifier. In section 3, an experimental scenario and the results of performance evaluation are presented. We conclude in section 4.

2 Detection Method

2.1 Feature Set Selection

In this paper, detecting DDoS attack is formulated as a classification problem. Features demonstrating the characteristics of the attacks are used as the input of the neural network classifier and the classifier already trained can judge whether attacks exist in the unseen data. The accuracy of neural network classifier depends largely on whether the selected features can really summarize the characteristics of DDoS attack. After a careful analysis of the various DNS DDoS attacks, 8 features are extracted from the DNS query data, which can to some extent separately or jointly reflect the DDoS attack. The features used as the input are:

- Query rate (QR). This feature denotes the number of queries per second the DNS server receives.
- Standard deviation of query volume ($SDQV$). Standard deviation of QRs during a certain period can be used to analyze changes in the query volume.
- IP address space ($IPAS$). $IPAS$ denotes the number of unique IP addresses observed during a period. In the period of the attack, the value of $IPAS$ may change greatly.
- Domain name space ($DNSP$). $DNSP$ is the number of unique domain names. In an attack situation, some of the malicious scriptings implementing the DDoS attack randomly generate domain names quickly and continually send query packets to the victim server for these spurious names.
- Number of queries issued from port 53 ($NP53$). Some famous DDoS applications send packets via port 53.
- Entropy of query types (EQT). Entropy, also known as *Mean Information Content*, is a physical quantity related to the degree of randomness.

Under abnormal circumstances the corresponding entropy of query types may change.

- Proportion of Recursive Query (*PRQ*). Enabling the recursion desired bit when flooding to a DNS authoritative server somewhat make the impact of attacks expanded.
- Average Length of Domain Names (*ALDN*). Some DDoS attacks request randomly generated domain names. The generating methods often start from fewer characters and then gradually increase the length. Therefore, the average length of domain names would be a useful indicator.

Based on these features the input vector with 8 elements is formulated: $I(t) = [QR(t), SDQV(t), IPAS(t), DNSP(t), NP53(t), EQT(t), PRQ(t), ALDN(t)]$, where t denotes the index of the time window.

2.2 Utilization of Neural Network

In most cases, the network consisting of two layers of adaptive weights with full connectivity is capable of approximating to arbitrary accuracy any continuous function from a compact region of input space, provided the the number of hidden units is sufficiently large and the weights and biases are chosen appropriately [5]. So we choose to use a feed-forward network with two layers: a hidden and a output layer.

According to *Kolmogorov's theorem* [6],if the input is constructed with R linear neurons, the hidden layer should has at least $2 \times R + 1$ neurons. So, hidden layer here is constructed with 17 sigmoid neurons. The output layer is made of one sigmoid neuron that generates values between 0 and 1. We represent a normal state by 0 and an attack state by 1. The back propagation algorithm based on *Scaled Conjugate Gradient* method [7] is used for training because of its fast training ability and sigmoid transfer function use *tansig*.

3 Experiments

3.1 Experimental Settings

An experimental scenario is set up in our local network to simulate DDoS attacks. In this scenario, there are 12 query clients, a control host and a DNS authoritative name server. Six of the clients called user clients automatically send normal DNS queries to the server and the server responds normally. The other 6 clients called zombie clients are implanted the malicious script in advance. We use the control host as an attacker to remote control the zombie clients which follow our commands and run the malicious script in them to send utmost possible number of malicious queries to the server. These queries in whole or in part follow the flow and query patterns we mentioned in section 2.1. The simulation of DDoS attack lasts for a couple of days and the malicious queries are sent continuously to the server lasting 5.4 hours. Details are given in Table 1.

Table 1. The data description of the simulated attack

Query State	Duration(min)	Query Rate(qps)
DDoS	324	20000~35000
Normal	2556	6000~20000

3.2 Performance Evaluation Criteria

Two criteria are chosen for evaluating performance of the classifier: *True Positive Rate (TPR)* and *False Positive Rate (FPR)*.

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{TN + FP} \quad (3)$$

In formula (3), *TP*(*True Positive*), *FN*(*False Negative*), *FP*(*False Positive*) and *TN* (*True Negative*) are defined in Fig. 3. *TPR* describes the sensitivity of our classifier while *FPR* shows the rate of false alarms. According to *TPR* and *FPR*, a *Receiver Operating Characteristic (ROC)* curve can be drawn, which is from signal detection theory.

In addition, two error evaluation criteria, *Mean Squared Error (MSE)* and *Percent Error (%E)* are also used:

- *MSE*: the average squared difference between outputs and targets.
- *%E*: indicates the fraction of samples which are misclassified.

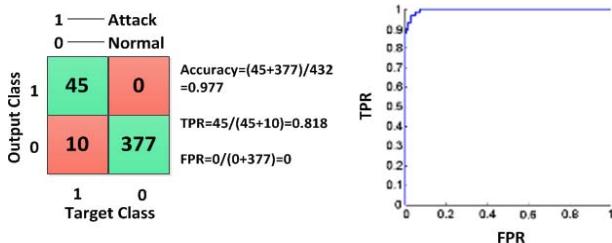
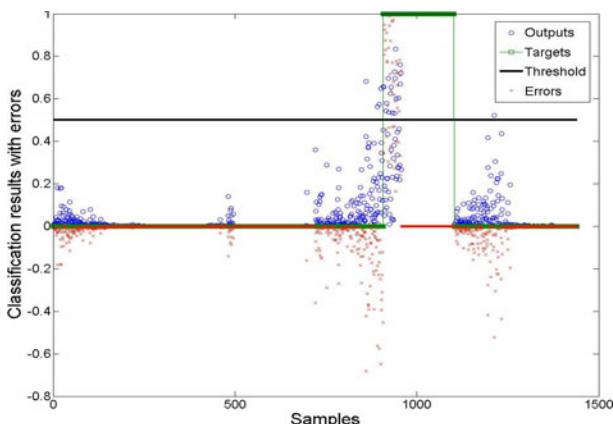
3.3 Result on Synthetic and Real Data

Firstly, the samples are divided into 3 data sets: training set (70%), validation set (15%) and test set (15%). Training set is presented to the network during training and the network is adjusted according to the training error. Validation set is used to measure the network generalization, and halt training when generalization stops improving. Test set has no effect on training and therefore provides an independent measure of the network performance. During the training phase, the best validation performance occurs at the 39th iteration and the network at this iteration is returned. The corresponding values of the errors are given in Table 2. To further analyze the performance of the classification, a test confusion matrix and an *ROC* curve are shown in Fig. 11.

In this figure, the green cells on the diagonal of the test confusion matrix (left) give the number of cases that are correctly classified, and the red cells show the misclassified ones. The *ROC* curve of testing results (right) is a plot of the true positive rate (*sensitivity*) versus the false positive rate (1-*specificity*). A perfect test would show points in the upper-left corner, with 100% *sensitivity* and 100% *specificity*. The confusion matrix shows fairly good recognition of attacks and normal states with *TPR* > 81% and *FPR* equal to 0. From the *ROC* curve, we can also see that the network performs well. In addition to the synthetic

Table 2. Minimum Errors corresponding to the three data sets

Data Set	Samples(min)	MSE	%E
Training	2,016	1.28986e-2	1.68650e-0
Validation	432	1.29352e-2	1.62037e-0
Test	432	1.74391e-2	2.31481e-0

**Fig. 1.** Confusion matrix and *ROC* curve of simulation results**Fig. 2.** Outputs of the neural network with errors

data obtained from experiment scenario, two days' query data from ".CN" DNS authoritative server are also applied to evaluate the performance. During this period the server had suffered DDoS Attack for more than 3 hours. One day's data are used to train and another day's to test. The detection performance of the neural network is visualized in Fig. 2. In this figure, outputs are represented by circles and targets by squares. The horizontal line represents the threshold of 0.5 above which data points are classified as attacks. Error between outputs and targets are denoted by the crosses. We can see from this figure that there are some output points below the threshold even though their target values are 1, which means some attacks are misclassified as the normal queries by the

classifier. In fact, it turns out that the $TPR = \frac{145}{145+50} \times 100\% = 74.7\%$, the FPR is 0 and the total accuracy is 96.5%.

4 Conclusion

In this paper we formulated the attack detection on DNS as a dualistic classification problem and designed a detection system against DDoS attack using a multi-layer feed-forward neural network classifier. Several features indicating the abnormal characteristics during DDoS attack were extracted from the DNS query flow and fed into the neural network classifier as an input. A careful assessment showed that two-layers neural networks applied in DDoS attack detection area does have a relatively good sensitivity and low false alarm rate.

Computational efficiency depends on the number of epochs of training, which can be exponential to the number of inputs in the worst case. However, there are a number of methods to accelerate training. For example, *Simulated Annealing* algorithm [9] not only accelerates training but also ensures the convergence to the global optimum. When the neural network parameters were determined by training, classification of a single input was done in a negligible time with $N^2 + M^2$ multiplication where N is the number of input features and M is the number of hidden neurons. Therefore, the neural network based detection method is capable of working as an online classifier.

References

1. Estan, C., Savage, S., Varghese, G.: Automatically inferring patterns of resource consumption in network traffic. In: Proceedings of the ACM SIGCOMM Conference (2003)
2. Rastegari, S., Saripan, M.I., Rasid, M.F.A.: Detection of Denial of Service Attacks against Domain Name System Using Neural Networks. IJCSI International Journal of Computer Science Issues 6(1) (2009)
3. Ryan, J., Lin, M., Miikkulainen, R.: Intrusion Detection with Neural Networks. In: AI Approaches to Fraud Detection and Risk Management: Papers from the 1997 AAAI Workshop, Providence, RI, pp. 72-79 (1997)
4. Cunningham, R., Lippmann, R.: Improving intrusion detection performance using keyword selection and neural networks. In: Proceedings of the International Symposium on Recent Advances in Intrusion Detection, Purdue, IN (1999)
5. Nabney, I.T.: NETLAB: Algorithms for Pattern Recognition, pp. 148–151. Springer, New York (2001)
6. Kolmogorov, A.N.: On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition (1957)
7. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6(4), 520–540 (1993)
8. Tanner Jr., W.P., Swets, J.A.: A decision-making theory of visual detection. Psychological Review 61(6), 401–409 (1954)
9. Ingber, L.: Simulated Annealing: Practice Versus Theory. J. of Mathematical and Computer Modeling 18(11), 29–57 (1993)

Classification Based on Multiple-Resolution Data View

Mateusz Kobos and Jacek Mańdziuk

Warsaw University of Technology,
Faculty of Mathematics and Information Science,
Plac Politechniki 1, 00-661 Warsaw, Poland
{M.Kobos,J.Mandziuk}@mini.pw.edu.pl

Abstract. We examine efficacy of a classifier based on average of kernel density estimators; each estimator corresponds to a different data “resolution”. Parameters of the estimators are adjusted to minimize the classification error. We propose properties of the data for which our algorithm should yield better results than the basic version of the method. Next, we generate data with postulated properties and conduct numerical experiments. Analysis of the results shows potential advantage of the new algorithm when compared with the baseline classifier.

Keywords: kernel density estimation, classification based on density estimation, average of density estimators.

1 Introduction

The main idea that inspired the work on the topic presented in this paper was that combining views on the data with different “resolutions” should produce an insight into the structure of the data. This insight, in turn, should aid solving problems related to data analysis such as classification. To implement the idea of data “resolution”, we have decided to use Kernel Density Estimator (KDE) with its bandwidth parameter interpreted as the “resolution”. In density estimation function generated by KDE, it can generally be observed that the larger (smaller) the bandwidth, the more similar (dissimilar) values at distant points. This phenomenon can be interpreted as manipulating the resolution. To combine different resolutions, we simply average output of KDEs with different bandwidths. Note that classifier based on KDE that we use was noticed by Specht in [1] to have the form of a neural network.

In practice, the above idea underlying the algorithm is implemented as follows. For a given test point and for each class, we use KDEs with different bandwidths to independently estimate density at the point. Then, for each class, the estimates are averaged to form final density estimate. Next, these estimates are inserted into the Bayes formula to produce probability estimate for each class. The bandwidths used in this process are selected beforehand to minimize the cross-validation classification error on the training set during the training

process. The minimization is carried out by the L-BFGS-B quasi-Newton optimization algorithm introduced in [2]. For a detailed description of the training and classification phases as well as an explanation of some of the design decisions made while developing the algorithm (e.g. why we are using a single bandwidth parameter per KDE instead of a matrix of parameters) see [3, Sect. 2].

The methods appearing in the literature that seem to be the most similar to the proposed algorithm build upon an idea of combining different density estimators. These approaches can be mostly divided into two groups. The first one embraces methods that use a combination of density estimators, and their main goal is to optimize quality of density estimation. Even if they are used in a classification task, the classification error is not optimized directly what generally should result in obtaining suboptimal classification outcome. This group includes the algorithm that uses a linear combination of Gaussian Mixture Models (GMMs) and KDEs with predefined bandwidths to estimate density [4]. Another example is the method in which the boosting algorithm is used on KDEs with fixed bandwidths [5]. Yet another approach is developed in [6] where the EM algorithm is used on an average of GMMs to optimize density estimation quality. In the other group, there are methods where a combination of classifiers based on density estimators is used. Classification error is optimized directly in these approaches; however, the algorithms are combined on classifiers level instead of being combined on a deeper level of density estimators. An example of the algorithm in this group is the BoostKDC binary classifier, proposed in [7], which uses Real AdaBoost method with classifiers based on KDEs.

We propose an algorithm that is situated between the above-mentioned groups; namely, it combines different density estimators, but the parameters of the estimators are selected directly to optimize the classification error (and not the quality of density estimation). Such an approach is quite novel. To authors' knowledge, the only other algorithm that belongs to this category is a binary classifier introduced in [8]. Our approach is significantly different from the mentioned algorithm and much simpler.

The efficacy of our algorithm was tested using 19 popular benchmark data sets. The goal of the experiments was to test whether using as little as two different KDEs improves real-world results of KDEs-based classifier and gives an algorithm that is competitive when compared with the methods from the literature. A detailed description of these experiments can be found in [3, Sect. 3], but the results can be recapitulated as follows. The introduced algorithm yielded statistically significantly better result than the baseline version. What is more, the comparison between the results obtained and the literature results indicates that the algorithm is competitive when compared with other classification methods.

In this article, we examine properties of the algorithm introduced in [9] and [3]. In Sect. 2, we hypothesize about properties of data sets for which our algorithm should produce superior results when compared with a baseline KDE-based classifier. To confirm our hypotheses, we generate an artificial dataset with postulated properties and test our algorithm on it in Sect. 3.

2 Data Characteristics

Let us consider a classification problem described by densities of two classes, each density defined by a four-element, two-dimensional Gaussian Mixture Model. We define two structures separated by a long distance. The first one is a large low-density structure while the second one is a small high-density structure. Both of them generate non-trivial optimal decision boundaries (see Fig. 1). In the case of the low-density structure, a KDE with a large bandwidth would model well the decision boundary, and in the case of high-density structure, a small bandwidth would be appropriate. An average of these two KDEs should also give good results since the KDE with the small bandwidth will dominate in the high-density region while the KDE with the large bandwidth will dominate in the low-density region.

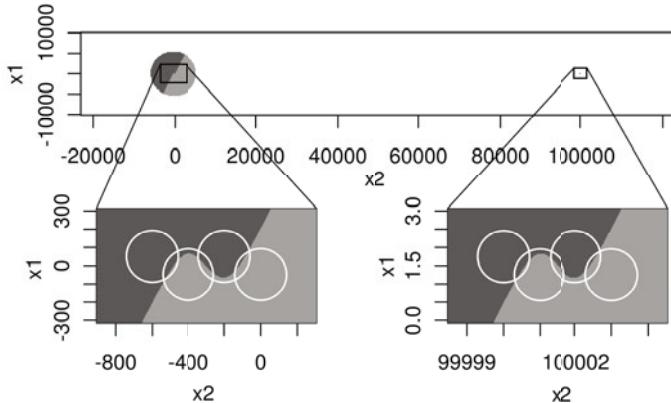


Fig. 1. Optimal decision regions (*dark gray* and *light gray* colors) in artificial dataset with a general view of the domain (*top*) and enlarged regions of low (*bottom left*) and high (*bottom right*) density structures. Note that the scale in depiction of both structures is different. Densities in each class are generated by a Gaussian Mixture Model (*white circles*). The white area in the general view of the domain corresponds to points where the density of both classes is negligibly small i.e. smaller than computer's machine precision.

The long distance that separates the structures is introduced for purely technical reasons. Our aim was to define a problem for which the optimal results are obtained in the case of equal bandwidths for both classes. This way we can narrow down the search space. Additionally, in the case of two bandwidths per class ($E = 2$), the misclassification error function depends on two parameters only; thus, it can be easily visualized. Both introduced structures are distant enough to be considered as separated because at every point, the density generated by at least one of the structures is negligibly small (smaller than the computer's machine precision). Therefore, they can be analyzed independently. Additionally, in each of the structures, the distribution of one class is a shifted version of

the other class's distribution. In this case, the use of a common bandwidth for both classes is justified (c.f. [10, p. 461]).

3 Experiments

The algorithm with two bandwidths ($E = 2$) per class was experimentally compared with the basic version of one bandwidth ($E = 1$) per class in the best-case scenario where the bandwidths were chosen optimally for the given classification problem. We generated a few instances of the problem, each having a different training set size. For each training set size, we generated independently 20 training sets, one testing set of size of 20 000 samples, and a set of bandwidth pairs. Each bandwidth came from the same range that starts at 0 and is long enough to contain the optimal bandwidth values for $E = 2$ and $E = 1$ cases (the length was selected manually). 100 equidistant bandwidth values were selected from this range to form a total number of 10 000 combinations of two bandwidths. A sample mean misclassification probability and a sample mean MSE were computed for the model based on each bandwidth pair. The mean values were computed over 20 training sets.

Figure 2 presents error functions computed for one of the training set sizes. Note that the values of the error function attainable for the $E = 1$ version belong to the half-line (a, a) , $a \in [0, \infty)$. This half-line does not have to contain the global minimum of $E = 2$ version, thus using more than one KDE can potentially improve the results. Indeed, this is the case in the example presented in Fig. 2, where the global minima are located far away from the above-mentioned half-line. Another interesting observation is that in this case, the optimal solution consists of a small and a large bandwidth; this is consistent with our expectations. A surprising fact is that the minimum of the mean MSE function is located very close to the minimum of the mean misclassification probability; the situation is similar for other training set sizes. This property does not have to hold in a general case (see e.g. [10, p. 459]), but here it additionally justifies our approach to bandwidths selection where we minimize MSE instead of directly minimizing the misclassification probability (see [3, Sect. 2-E] for details).

The main objective of the experiment was to compare the best-case scenario results of $E = 1$ version of the algorithm with $E = 2$ version using training data sets of different sizes. Analysis of these results (see Fig. 3) leads to interesting observations. The most important one is that for each data set size except of the smallest one, the $E = 2$ version yields results that are statistically significantly better (paired t-test, $p \leq 1.56 \cdot 10^{-9}$) by a large margin than the results of the $E = 1$ version. For the larger data sets (800 samples and more), the $E = 2$ version yields misclassification error that is approximately two times smaller than the error of the $E = 1$ version when normalized by subtracting the Bayes risk. Moreover, in the case of the $E = 2$ version, minimizing the mean MSE gives results that are not statistically significantly different (paired t-test, $p \geq 0.142$) from minimizing the mean misclassification probability.

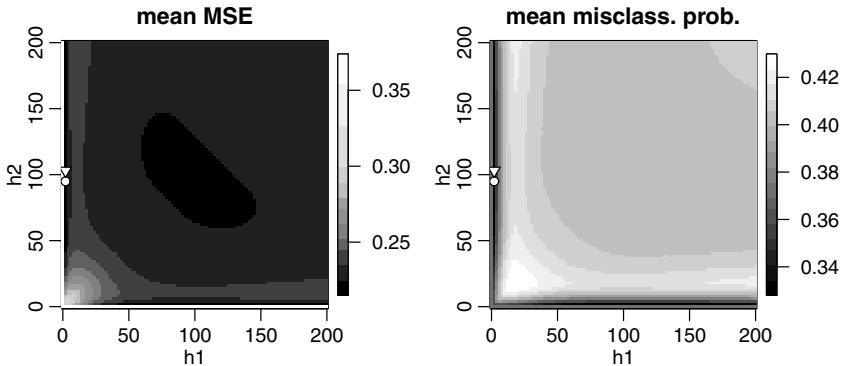


Fig. 2. Sample mean MSE (*left*) and sample mean misclassification probability (*right*) computed on the testing data from the artificial data set with the training set size equal 400. The darker the color of a point, the smaller the function value. The axes correspond to bandwidth values for each KDE. Note that in both plots, the points where one of the coordinates equals 0 correspond to high values of the function. The global minima of the mean misclassification probability and mean MSE are marked with a triangle and a circle respectively.

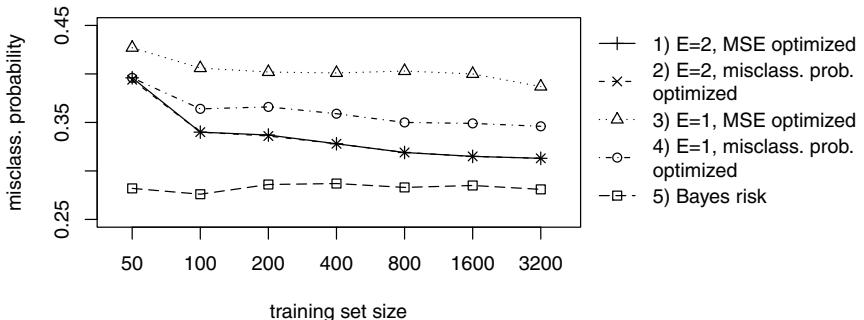


Fig. 3. Average misclassification probabilities on the artificial data set for different versions of the algorithm with optimally chosen bandwidths. Examined algorithm versions: 1) $E = 2$ with mean MSE optimized, 2) $E = 2$ with mean misclassification error optimized directly, 3) $E = 1$ with mean MSE optimized, 4) $E = 1$ with mean misclassification error optimized directly. Additionally, a sample optimal Bayes risk rate is also showed - line 5).

In summary, the proposed algorithm's version ($E = 2$) gives better best-case scenario results than the basic version ($E = 1$), which is consistent with our expectations. Additionally, for the data set considered and $E = 2$ version, minimizing the MSE gives results that are as good as the ones obtained by minimizing directly the misclassification probability.

4 Conclusions and Future Work

We showed that best-case scenario results on an artificial data set yielded by our algorithm are better than those yielded by the basic version. Currently, we are working on a version of the algorithm where the number of bandwidths is adjusted automatically to the data. This way no “expert knowledge” is required to choose the number of bandwidths per class E for a classification problem at hand. Another modification worth testing is selecting the starting point in a different way (e.g. by Scott’s normal reference rule or Sheather-Jones method (see [11, Sect. 3])).

Acknowledgments. The authors would like to thank Prof. Jacek Koronacki for valuable suggestions concerning the direction of this research. This work has been supported by the European Union in the framework of European Social Fund through the Warsaw University of Technology Development Programme, by the European Social Fund and the National Budget in the framework of Integrated Operational Programme for Regional Development (ZPORR), Action 2.6: “Regional Innovation Strategies and Transfer of the Knowledge” through the Mazovian Voivodeship “Mazovian PhD Student Scholarship”, and by the Warsaw University of Technology research grant.

References

1. Specht, D.: Probabilistic neural networks. *Neural Networks* 3, 109–118 (1990)
2. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Stat. Comput.* 16, 1190–1208 (1995)
3. Kobos, M.: Combination of independent kernel density estimators in classification. In: Ganzha, M., Paprzycki, M. (eds.) International Multiconference on Computer Science and Information Technology, vol. 4, pp. 57–63 (2009)
4. Smyth, P., Wolpert, D.: Linearly combining density estimators via stacking. *Mach. Learn.* 36, 59–83 (1999)
5. Di Marzio, M., Taylor, C.C.: Boosting kernel density estimates: A bias reduction technique? *Biometrika* 91, 226–233 (2004)
6. Ormoneit, D., Tresp, V.: Averaging, maximum penalized likelihood and bayesian estimation for improving gaussian mixture probability density estimates. *IEEE T. Neural. Netw.* 9, 639–650 (1998)
7. Di Marzio, M., Taylor, C.C.: On boosting kernel density methods for multivariate data: density estimation and classification. *Stat. Methods Appl.* 14, 163–178 (2005)
8. Ghosh, A.K., Chaudhuri, P., Sengupta, D.: Classification using kernel density estimates: Multiscale analysis and visualization. *Technometrics* 48, 120–132 (2006)
9. Kobos, M., Mańdziuk, J.: Classification based on combination of kernel density estimators. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009. LNCS, vol. 5769, pp. 125–134. Springer, Heidelberg (2009)
10. Ghosh, A.K., Chaudhuri, P.: Optimal smoothing in kernel discriminant analysis. *Stat. Sinica* 14, 457–483 (2004)
11. Wand, M.P., Jones, M.C.: Kernel Smoothing. Chapman and Hall, London (1995)

Identification of the Head-and-Shoulders Technical Analysis Pattern with Neural Networks

Achilleas Zapranis and Prodromos Tsinaslanidis

Department of Accounting and Finance, University of Macedonia of Economic and Social Sciences, P.O. Box 1591, 54006 Thessaloniki, Greece
`{Zapranis, tsinpro}@uom.gr`

Abstract. In this paper we present a novel approach for identifying the head-and-shoulders technical analysis pattern based on neural networks. For training the network we use actual patterns that were identified in stochastically simulated price series by means of a rule-based algorithm. Then the patterns are being converted to binary images, in a manner similar to the one used in hand-written character and digit recognition. Our approach is tested on new simulated price series using a rolling window of variable size. The results are very promising with an overall correct classification rate of 97.1%.

Keywords: Pattern Recognition, Technical Analysis, Head-and-Shoulders, Neural Networks, Identification.

1 Introduction

In this paper we look at the pattern recognition problem from a financial perspective. To be more precise a multiple layer feed-forward neural network is incorporated for identifying within simulated price series the Head-and-Shoulders (HS) pattern. HS is one of the most known price pattern of technical analysis (TA). The preparation of the training dataset is being done with methods similar to those used in hand-written character and digit recognition. Patterns of TA include a high level of subjectivity. This makes in practice the pattern “unobservable”. Usually technical patterns are identified by rule-based mechanisms [1]. One way to train the network would be with cases where HS pattern was identified by known investment advisors, or other financial intermediaries. Such a procedure would be time-consuming and impractical. In this paper, the Geometric Brownian Motion (GBM) is used to simulate an adequate number of simulated stochastic price series. After that, actual patterns identified in stochastically simulated price series by means of a rule-based algorithm are used to train the network. Before training the network these patterns are converted into binary images, then into sub-total matrixes, and finally into column vectors. The network is tested on new simulated price series using a rolling window of variable size (‘multi-sized rolling window’). The network seems to perform near perfect with an overall classification rate of 97.1%.

The remainder of the paper is organized as follows. Section 2 describes the Technical analysis and the HS pattern. Section 3 describes how neural networks deal with

pattern recognition challenges. Section 4 describes the input data preparation. Section 5 describes the creation of the multi-layer feed-forward network, and section 6 concludes.

2 Technical Analysis and the Head-and-Shoulders Pattern

Some see TA as a powerful tool in order to predict future prices and returns of stocks and other investment products. According to [2] “*technical analysis is the science of recording, usually in graphic form, the actual history of trading (price changes, volume of transactions, etc.) in a certain stock or in “the Averages” and then deducing from that pictured history the probable future trend*”. The main tools of TA are indicators and patterns. The later are distinctive formations created by the movements of security’s prices on a chart. Two well known technical patterns are the HS and the Cup-with-Handle. More information on TA indicators and chart patterns can be found in [2, 3].

The HS pattern’s name comes from its resemblance of the upper part of human body. It consists of three peaks, with two intervening troughs between the head and each shoulder. The neckline is drawn through the lowest points of these intervening troughs and may slope upward or downward. When the price breaks downwards the neckline, a fall of the price equal to the distance between the head and the neckline (head’s height) is expected. There are four categories of the HS pattern. The distinction is based on the trend (bull or bear trend) before the pattern identified, and on the pattern’s form (normal or inverse). When the HS pattern has its normal form we expect a down-trend when the “neckline” is crossed. Reversely, when it has the inverse form we expect an up-trend. When the normal form occurs and an up-trend preexists the pattern is characterized as a trend-reversal pattern. Down-trend preexistence characterizes the pattern as trend-continuation. Similarly, when the inverse form takes place the pattern is characterized as a trend-reversal or as a trend-continuation when a down-trend or an up-trend preexists respectively.

3 Neural Networks and Pattern Recognition

The concept of neural networks started as an effort to describe the human brain behavior. An artificial neural network is a system emulating a biological neural system. Artificial neural networks emerged after the introduction of simplified neurons [4]. Today’s architectures are able to make systems that perform similar with the human brain. Pattern recognition is one of the main actions performed by the human brain. Personal identification from handwriting, voice, fingerprints, facial images etc. can be achieved by identifying patterns hidden in biological data. Hand-written character recognition [5-9] and digit recognition [10] are well-known pattern recognition challenges.

The concept of *classification* involves the learning of likenesses and differences of objects in a population of no identical objects [11]. Pattern *recognition* is the process defining that an object from a population P belongs to a known subpopulation S. The recognition of an object as a unique singleton class is called *identification*. On the

other hand, classification is the process of grouping objects into classes according to their similarities. Recognition and classification are both included in pattern recognition. Additionally, [11] presents a taxonomy of pattern recognition methods. The three main classes of pattern recognition methods are:

- 1) *Decision-theoretic methods* which consist of statistical, graph-theoretic, and rule-based methods; [5], [1].
- 2) *Structural/Syntactic methods* which consist of automata, Hopfield recurrent neural networks and bidirectional associative maps.
- 3) *Associative mappings (neural and fuzzy mappings)* which consist of feed-forward neural networks, self-organizing networks and hybrid networks;[6, 7, 10, 12-14].

In our network, the training patterns were preprocessed with methods similar to those used in handwritten character and digit recognition. In [10], a combination of Self Organizing Maps (SOMs) with fuzzy rules was applied for handwritten numerical recognition. The training patterns were rescaled and centered onto a 64×64 binary grid. This method is also adopted in this paper at the data preprocessing stage where the HS patterns identified in stochastic price series were converted into 64×64 binary matrixes before being used as inputs for the network. Similar methods were used in many studies in the field of hand-written character recognition; [5] and [12].

4 Data Preparation

The HS is one of the most known patterns in TA. However, it is considered to occur very seldom. As a result very few verified time series exist and a satisfactory large training dataset cannot be created in order to use it as an input to the feed-forward neural network. Simulated data is used to rectify this. More precisely, the GBM¹ is used to create a large number of simulated price paths. The GBM is given by:

$$dS = S\mu dt + S\sigma\varepsilon\sqrt{dt} \quad (1)$$

where μ is the expected return, σ is the volatility of the price series, dt is a short interval of time and ε follows a standard normal distribution. GBM is considered as an accurate representation of the underlying stock price (S) generation mechanism.

First, we construct 250,000 price paths of 500 observations each with different combinations of drift rate and volatility rate. On each price series a rule-based mechanism was applied to identify the HS pattern by applying the criteria presented in [1]. The rule-based algorithm identifies the HS pattern in a price path and returns the characteristic points (local peaks and troughs) of the pattern. Cases where the pattern is identified or not, are categorized into two groups in which the target pattern is assigned to value 1 or 0 respectively.

However, the rule-based mechanism based on the criteria presented in [1] is limited and can identify only the pattern that has the normal form and uptrend preexistence. On the other hand, in our proposed methodology the network is trained to identify all four cases mentioned in section 2. To do so, after the rule based script returns the characteristic points (locals) of the pattern (Fig. 1), the points of the trend

¹ For further details about the Geometric Brownian Motion see [15].

preexistence (P_0 and T_0) are excluded and the intersection between the neckline and the uptrend that preexists (P^{**}) is added. In other words, from the initial formation only the part above the neckline remains. For the identification of the HS pattern's inverted form, the network can reexamine the same price series after multiplying the series with minus one.

In order for all the inputs of the network to have the same formation, the method mentioned in [10] is applied. Each part of the price series is converted into a binary 64×64 matrix. This can be achieved by scaling the prices and the days of the price series between 1 and 64 and rounding them toward the closest integer. This results in the coordinates which prices are represented by ones on the 64×64 matrix. The value of zero is assigned to the remaining prices of the matrix. For the improvement of the network the initial binary matrix is bordered into 4×4 sub-blocks resulting in a 16×16 matrix where each point expresses the sum of each sub-block. So each point of the latest matrix has values between 0 and 16. Finally, the matrix is converted into a vector ready to be used as an input to the network.

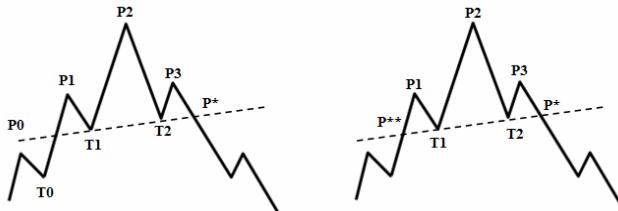


Fig. 1. The locals of the trend reversal HS pattern with the normal form. On the left pattern are illustrated the characteristic points of the pattern given by the rule based mechanism. On the right pattern only the locals that are used as inputs of the network, are presented.

The network, after the training stage will try to identify the pattern in new price series at different time intervals. This will have an impact in the 64×64 binary matrix. For larger period of times after the scaling of the price series there will be more points in the matrix with the value of one. In other words, this will affect the thickness of the price series when viewed in the binary matrix. To overcome this problem the network is trained with inputs of different thicknesses.

Before the scaling, a linear connection between the characteristic points of the pattern is applied. By this, the noise between those locals is extracted and the main structure of the pattern is isolated without taking into account the price's fluctuation between the locals. The thickness parameter mentioned before helps to deal with the noise realized at the price series when the network is used for identification.

Another key point to the preparation of the data is the slope of the neckline that can vary. We exclude 30% of the cases with the highest and another 30% of the cases with the lowest necklines' slopes, in order to include more symmetrical HS patterns in our training dataset. This is an addition to the criterion presented in [1].

For the network's training, two thousands different input vectors are used. Half targeted with ones, which represent HS pattern cases, and the other half targeted

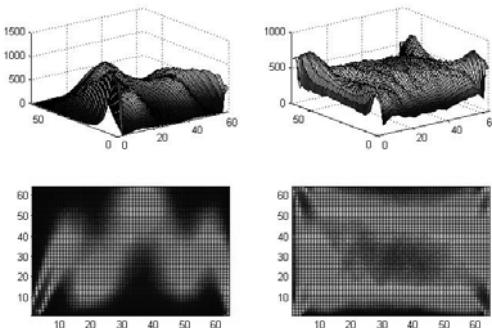


Fig. 2. Top sub-figures; cumulative, three dimensional distributions of the input binary matrixes. Down sub-figures; ground plans of the above 3D sub-figures. Left sub-figures; Inputs targeted with ones (HS patterns). Right sub-figures; Inputs targeted with zeros (no HS patterns).

with zeros that represent cases where the pattern was not identified. Fig. 2 illustrates cumulative, three dimensional (3D) distributions of the input binary matrixes before they are converted into column vectors. Left sub-figures present the inputs targeted with one where the pattern is clear, and right sub-figures the inputs with target zero where a very noisy picture can be seen. The top sub-figures show a three dimensional illustration while the bottom ones show a ground plan of the above 3D figures. These figures help illustrate the inputs used to feed the network.

5 The Neural Network

A two-layer feed-forward network, with tan-sigmoid transfer functions in both the hidden layer and the output layer can classify vectors sufficiently, given enough neurons in its hidden layer. The network's aim is to decide whether a specific price series is a HS pattern or not. For that reason, the network has two output neurons because there are two categories associated with each input vector. With the Neural Network Pattern Recognition Tool (*nprtool*) it is possible for the user to create, train a network, and evaluate its performance using mean square error and confusion matrices².

The Scaled Conjugate Gradient algorithm is used for training. 60% of the input patterns is used for training, 20% is used to validate that the network is generalizing and to stop training before overfitting, and 20% is used as an independent test of network generalization.

In tables 1-4 the confusion matrices for training, testing, validation and an overall view are illustrated. Each table consist of three columns (a, b and c) and three rows (1, 2 and 3). The network's outputs are near perfect, as shown by the high numbers of correct responses in the bordered cells (cells a1 and b2) and the low numbers of incorrect responses in the not bordered cells (cells a2 and b1). The lower right bold percentages (cell c3) illustrate the overall accuracies. Results of the training, validation, and test state indicate that the network is able to classify the input data with great

² For further details about *nprtool* see the product help of *Matlab*. 2009b. The Mathworks, I.

accuracy. This is confirmed by the Receiver Operating Characteristic (ROC) curve which indicates that the network performs near perfectly.

The next step is to apply a rolling window of variable length on the price series and then convert the parts of the series into forms similar to the ones mentioned in the training stage. Variable length is used to capture all possible sizes of the pattern. First, the rolling window has the same size as the price series. Then the window is halved, and rolls with a step of 5 days³. By incorporating this “divide and roll” procedure the network seeks in the price series the HS pattern with different views. It can be argued that a technical analyst does the same, by focusing on the series at different view levels. Overall, our network identifies most patterns found by the rule-based mechanism.

Table 1. Training Confusion Matrix

Output Class	1	597	9	98.5%
	2	49.8%	0.8%	1.5%
2	6	588	99.0%	
	0.5%	49.0%	1.0%	
	99.0%	98.5%	98.8%	
	1.0%	1.5%	1.2%	
	1	2		
	Target Class			

Table 2. Validation Confusion Matrix

Output Class	1	192	18	91.4%
	2	48.0%	4.5%	8.6%
2	6	184	184	96.8%
	1.5%	46.0%	3.2%	
	97.0%	91.1%	94.0%	
	3.0%	8.9%	6.0%	
	1	2		
	Target Class			

Table 3. Test Confusion Matrix

Output Class	1	194	14	93.3%
	2	48.5%	3.5%	6.7%
2	5	187	97.4%	
	1.3%	46.8%	2.6%	
	97.5%	93.0%	95.3%	
	2.5%	7.0%	4.7%	
	1	2		
	Target Class			

Table 4. All Confusion Matrix

Output Class	1	983	41	96.0%
	2	49.1%	2.1%	4.0%
2	17	959	959	98.3%
	0.9%	47.9%	1.7%	
	98.3%	95.9%	97.1%	
	1.7%	4.1%	2.9%	
	1	2		
	Target Class			

6 Conclusion

The aim of this study is to create a neural network that identifies the HS technical price pattern in a price series of a stock. To prepare our training data we use a method similar to the ones presented in other hand-written character or digit recognition studies. We overcome the small observation frequency of the pattern with a rule-based pattern identification mechanism and a large number of simulated stochastic price series that represent the price generation mechanism of real stock prices. The network classified the input data near perfectly (97.1%) while at the same time outperformed the rule-based mechanism. More precisely, the neural network in contrast to the rule-based algorithm identifies all four different forms of the HS pattern. Our future

³ We prefer a 5-days step than a daily step for accelerating the network’s identification process.

studies will focus on the measurement of the network's identification power and determine statistically if it outperforms the rule-based mechanism. The aforementioned system can be used in the future as a tool for weak-form market efficiency⁴ test in financial markets.

References

1. Bernd, L.: Are Technical Trading Rules Profitable? Evidence for head-and-shoulder rules. *Applied Economics* 35, 33–40 (2003)
2. Edwards, R.D., Magee, J.: *Technical Analysis of Stock Trends*. 7th edn. (1997)
3. Achelis, S.B.: *Technical Analysis From A to Z* (1995)
4. McCulloch, W.S., Pitts, W.H.: A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115–133 (1943)
5. Li, T.F., Yu, S.S.: Handprinted Chinese character recognition using the probability distribution feature. *Int. J. Pattern Recognit. Artif. Intell.* 8(5), 1241–1258 (1994)
6. Tsay, M.-K., Shyo, K.-H., Chang, P.-C.: Feature Transformation with Generalized Learning Vector Quantization for Hand-Written Chinese Character Recognition. *IEICE Trans. Inf. & Syst.*, E82-D(3) (1999)
7. Camastra, F., Vinciarelli, A.: Cursive character recognition by learning vector quantization. *Pattern Recognition Letters* 22, 625–629 (2001)
8. Liu, C.-L., Nakagawa, M.: Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. *Pattern Recognition* 34, 601–615 (2001)
9. Liu, C.-L., Sako, H., Fujisawa, H.: Performance evaluation of pattern classifiers for handwritten character recognition. *International Journal on Document Analysis and Recognition* 4, 191–204 (2002)
10. Chi, Z., Wu, J., Yan, H.: Handwritten Numeral Recognition Using Self-Organizing Maps and Fuzzy Rules. *Pattern Recognition* 28(1), 59–66 (1995)
11. Looney, C.G.: *Pattern Recognition Using Neural Networks* (1997)
12. Tseng, D.C., Chiu, H.P., Cheng, J.H.: Invariant handwritten Chinese character recognition using fuzzy ring data. *Image and Vision Computing* 14, 647–657 (1996)
13. Kohonen, T.: Self-Organizing Maps. In: Huang, T.S., Kohonen, T., Schroeder, M.R. (eds.) 2nd edn. Springer series in information sciences, vol. 30 (1997)
14. Cho, S.-B.: Ensemble of structure-adaptive self-organizing maps for high performance classification. *Information Sciences* 123, 103–114 (1999)
15. Hull, J.C.: *Options, Futures, and Other Derivatives*. 6th edn. (2006)
16. Fama, E.: 'Efficient Capital Markets: A Review of Theory and Empirical Work'. *Journal of Finance* 25(2), 383–417 (1970)
17. Fama, E.: 'Efficient Capital Markets II'. *Journal of Finance* 46(5), 1575–1617 (1991)

⁴ For market efficiency hypothesis see [16], [17].

Analyzing Classification Methods in Multi-label Tasks

Araken M. Santos¹, Laura E.A. Santana², and Anne M. Canuto²

¹ Campus Angicos - Federal Rural University of Semi-Árido (UFERSA) Angicos,
RN - BRAZIL, 59515-000

² Informatics and Applied Mathematics Department - Federal University of Rio Grande do
Norte (UFRN) Natal, RN - BRAZIL, 59072-970
araken@ufersa.edu.br, lauraemmanuel@yahoo.com.br,
anne@dimap.ufrn.br

Abstract. Multi-label classification methods have been increasingly used in modern application, such as music categorization, functional genomics and semantic annotation of images. This paper presents a comparative analysis of some existing multi-label classification methods applied to different domains. The main aim of this analysis is to evaluate the performance of such methods in different tasks and using different evaluation metrics.

1 Introduction

Multi-label classification was originally motivated by tasks on the contexts of text categorization and medical diagnosis [1,2]. However, multi-label classification has attracted significant attention from researchers lately, motivated from an increasing number of different applications, such as semantic annotation [3,4], music categorization [8], directed marketing [9], among others. Although there are a reasonable number of multi-label classification methods proposed in the literature, there is little effort in comparing the different multi-label methods in different applications. In [2], for instance, it was presented a comparative analysis of some existing methods applied to the protein domain. Nonetheless, with increasing number of possible multi-label applications in different domains, it is important to perform a broader comparison, using different application domains. As a contribution to this important topic, this paper presents a comparative analysis of some existing multi-label classification methods, using datasets of different domains. In order to do this analysis, these methods (Binary Relevance (BR), Label Powerset (LP) and Random k-labelsets (RAkEL)) will be applied using different traditional classification methods and will be evaluated using 11 different evaluation metrics. As a result of this analysis, we aim to investigate these classification methods under different circumstances.

2 Multi-label Classification

In the literature, different methods have been proposed to be applied to multi-label classification problems, such as [1,3]. These methods can be broadly categorized into two groups [1]. The next two subsections will describe these two groups in more details.

2.1 Problem Transformation Methods

In this approach, the main idea is to transform the original multi-label problem into a set of single-label classification problems. It is an algorithm independent approach, since its functioning does not depend directly on the classification method used. Problem transformation approaches have employed classical algorithms. In this paper, we will compare the three most widely applied in the literature, which are:

- Binary Relevance (BR) is a popular and the most widely-used problem transformation method [1]. BR considers the prediction of each label as an independent binary classification task. Thus, BR builds M binary classifiers, one for each different label L ($M = |L|$). For the classification of a new instance, BR outputs the union of the labels l_i that are positively predicted by the M classifiers.
- Label Powerset (LP) is a simple and less common problem transformation method [1]. LP considers each unique set of labels that exists in a multi-label training set as one of the labels of a new single-label classification task. Given a new instance, the single-label classifier of LP outputs the most likely label, which is actually a set of labels. Its main drawback is that it suffers from the increasing complexity emerged from the large number of label subsets and the majority of these classes are associated with very few examples [10].
- Random k-labelsets (RAkEL) constructs an ensemble of LP classifiers. Each LP classifiers is trained using a different small random subset of the set of labels. An average decision is calculated for each label l_i in L , and the final decision is positive for a given label if the average decision is larger than a given threshold t . RAkEL aims to take into account label correlations and at the same time avoiding the aforementioned problems of LP [10].

2.2 Algorithm Adaptations Methods

In this approach, extensions of single-label classifiers are proposed, adapting their internal mechanisms to allow their use in multi-label problems. Furthermore, new algorithms can be developed specifically for multi-label problems [2]. In the literature, several algorithm adaptation methods are proposed, based in different algorithms, such as: decision trees, probabilistic methods, neural networks, support vector machines, lazy and associative methods, boosting, among others. In [5], the most popular decision tree algorithm, named C4.5, was adopted for the handling of multi-label data. In [6], the backpropagation algorithm was adapted to the multi-label learning. In this adaptation, named BP-MLL, a new error function is introduced that takes multiple labels into account. The ML-kNN [3] is one extension of the k-Nearest Neighbors (kNN) that uses the maximum a posteriori principle in order to determine the label set of the test instance, based on prior and posterior probabilities for the frequency of each label within the k neighbors.

3 Evaluation Metrics

The evaluation of multi-label classifiers requires different measures than those used in the case of single-label problems. Unlike the single-label problems, which the classification of an example is correct or incorrect, in a multi-label problem a classification

of an example may be partially correct or partially incorrect. This can happen when a classifier correctly assigns an example to at least one of the labels it belongs to, but does not assign to all labels it belongs to. Also, a classifier could also assign to an example to one or more labels it does not belong to [2].

Several measures have been proposed in the literature for the evaluation of multi-label classifiers. According to [1], these measures can be broadly categorized in two groups: bipartition-based and ranking-based. Some of the bipartition-based measures, called example-based-measures, evaluate bipartitions over all examples of the evaluation dataset. Other bipartition-based measures, named label-based measures, decompose the evaluation process into separate evaluations for each label. Furthermore, the ranking-based measures evaluate rankings with respect to the ground truth of multi-label dataset. In this paper, we will use 11 evaluation measures, where 6 were example-based measures (Hamming Loss, Precision, Accuracy, Recall, F-Measure and Subset Accuracy), 2 were label-based measures (Micro F1 and Macro F1) and 3 were ranking-based measures (One-Error, Coverage and Average Precision). For more details in evaluation measures, see [1,2].

4 Experimental Work

Three multi-label datasets were chosen (datasets available at <http://mlkd.csdl.auth.gr/multilabel.html>). The first one is Yeast, a biological dataset that is concerned with protein function classification [5]. The second dataset is Scene, an image dataset scene [3] that is concerned with semantic indexing of still scenes. This dataset contains 2407 images associated with up to 6 concepts, such as beach, mountain and field. The third dataset is emotions, a music emotions dataset [8] that is concerned with the classification of songs according to the emotions they evoke.

Table 1. Standard and multi-label statistics

Datasets	Example	Attributes		Labels	DLS	LC	LD
		NUM	DIS				
yeast	2417	103	0	14	198	4.327	0.302
scene	2712	294	0	6	15	1.074	0.179
emotions	593	72	0	6	27	1.868	0.311

Table 1 illustrates some basic statistics of these datasets, such as the number of numeric (NUM) and discrete (DIS) attributes, along with multi-label data statistics, such as the number of distinct label subsets (DLS), the label cardinality (LC) and the label density (LD) [1]. Label cardinality is the average number of labels per example, while label density is the same number divided by $|L|$.

4.1 Methods and Methodology

As already mentioned, three different multi-label classification methods will be used in this investigation. For each multi-label model, we apply five supervised learning algorithms, which are: k nearest neighbor (KNN), decision tree (DT), support vector machines (SVM), naïve bayes (NB), and multilayer perceptron (MLP). The experimental results were evaluated using 11 evaluation measures, where 6 were

example-based measures, 2 were label-based measures and 3 were ranking-based measures. All multi-label classification methods and supervised learning algorithms used in this work are implementations of the Weka-based package of Java classes for multi-label classification, called Mulan [7].

The experiments were conducted using the 10-fold cross-validation methodology. In order to compare the obtained from the different learning methods, a statistical test will be applied, which is called hypothesis test (t-test).

5 Results and Discussion

Table 2 shows the performance of the multi-label classification methods when applied to yeast dataset, using all five supervised learning algorithm. The results are presented using 11 different measures evaluation. The statistical test compared the best results (**bold**) with the results of the other learning methods, in a two-by-two basis. The results which are statistically significant are underlined.

Analyzing the performance of BR, it is seen that the k-NN delivered the best results in most of the cases (7 out of 11), followed by NB (2) and SVM (2). Using LP as multi-label method, SVM has reached the best performance in most of the cases (8 out of 11), followed by KNN (1), DT (1) and MLP (1). Unlike the other two methods, when using RAKEL, it is possible to observe that there is no predominance of the learning methods, since k-NN and NB delivered the best performance in only four cases and they are followed by SVM (2) and DT (1). The statistical test showed that the performances of the learning methods are similar, since the best results are statistically significant in a small number of cases (3 for BR, 1 for LP and 2 for RAKEL).

Table 2. Results of the classification methos using yeast dataset

	BR										
	HL↓	Acc↑	Prec↑	Rec↑	F-M↑	SA↑	MaF1↑	MiF1↑	OE↓	Cov↓	AP↑
kNN	0.193	0.522	<u>0.710</u>	0.602	0.652	0.201	0.652	0.402	<u>0.227</u>	6.359	0.188
DT	0.250	0.433	<u>0.599</u>	0.573	0.585	0.060	0.581	0.386	0.393	<u>9.337</u>	0.362
SVM	0.199	0.499	0.714	0.575	0.637	0.146	0.633	0.324	0.256	<u>9.096</u>	0.460
NB	0.301	0.420	<u>0.529</u>	0.612	0.568	0.093	0.548	0.451	0.346	<u>7.500</u>	0.256
MLP	0.239	0.458	0.617	0.590	0.603	0.107	0.597	0.438	0.290	<u>7.293</u>	<u>0.213</u>
	LP										
kNN	0.213	0.523	0.657	0.627	0.641	0.245	0.638	0.437	0.269	8.149	0.430
DT	0.279	0.412	0.542	0.541	0.541	0.135	0.540	0.386	0.343	<u>9.201</u>	0.542
SVM	0.206	0.530	0.667	0.621	0.643	0.260	0.643	0.418	0.267	8.065	0.426
NB	0.242	0.468	0.593	0.590	0.591	0.183	0.596	0.440	0.321	<u>8.335</u>	0.486
MLP	0.256	0.465	0.586	0.594	0.590	0.194	0.584	0.443	0.334	8.501	0.491
	RAKEL										
kNN	0.208	0.493	0.683	0.575	0.624	0.163	0.625	0.380	0.259	9.155	0.438
DT	0.252	0.429	0.592	0.561	0.576	0.083	0.573	0.356	0.337	<u>9.616</u>	0.408
SVM	0.207	0.487	0.690	0.571	0.625	0.128	0.624	0.333	0.255	9.273	0.442
NB	0.279	0.414	0.542	0.576	0.559	0.087	0.555	0.405	0.383	<u>9.320</u>	0.459
MLP	0.229	0.468	0.634	0.582	0.607	0.123	0.605	0.394	0.302	9.184	0.398

Table 3 shows the performance achieved when using the scene dataset. It is seen that, for BR, once again, k-NN presented the best results in most of the cases (8 out of 11), followed by SVM, NB and MLP (1 out of 11). Using LP, SVM delivered the best results in all cases. For RAKEL, unlike the previous dataset, it is possible to note a predominance of MLP for (7 out of 11), followed by k-NN (4 out of 11). Unlike the

Table 3. Results of the classification methos using Scene dataset

	BR										
	HL↓	Acc↑	Prec↑	Rec↑	F_M↑	SA↑	MaFI↑	MiFI↑	OE↓	Cov↓	AP↑
kNN	0.094	0.643	0.668	<u>0.645</u>	0.656	0.617	0.705	0.707	0.262	0.527	<u>0.098</u>
DT	0.134	0.539	<u>0.556</u>	<u>0.639</u>	0.595	0.429	0.626	0.637	0.404	<u>1.232</u>	0.298
SVM	0.106	0.594	<u>0.612</u>	<u>0.643</u>	0.627	0.529	0.682	0.688	<u>0.346</u>	0.990	0.377
NB	<u>0.241</u>	<u>0.452</u>	<u>0.460</u>	0.859	<u>0.599</u>	0.167	0.558	<u>0.576</u>	0.535	1.003	0.201
MLP	0.100	0.625	0.645	0.671	0.657	0.562	0.701	0.697	0.265	0.577	0.095
	LP										
kNN	0.097	0.713	0.744	0.714	0.729	0.682	0.720	0.728	0.256	0.904	0.814
DT	<u>0.147</u>	<u>0.580</u>	<u>0.601</u>	<u>0.602</u>	<u>0.602</u>	0.538	<u>0.591</u>	<u>0.601</u>	0.409	<u>1.159</u>	0.727
SVM	0.090	0.735	0.761	0.749	0.755	0.696	0.745	0.754	0.246	0.733	0.833
NB	0.137	0.614	0.630	0.680	0.654	0.531	0.638	0.646	0.403	1.049	0.742
MLP	0.106	0.695	0.721	0.713	0.717	0.651	0.701	0.705	0.283	<u>0.855</u>	0.806
	RAKEL										
kNN	0.095	0.694	0.724	0.698	0.710	0.662	0.720	0.726	0.257	<u>0.883</u>	0.816
DT	0.107	0.639	<u>0.660</u>	0.710	0.684	0.550	0.701	0.713	0.270	0.593	0.835
SVM	0.097	0.671	0.692	0.720	0.706	<u>0.602</u>	0.724	0.734	0.237	<u>0.637</u>	0.847
NB	<u>0.179</u>	<u>0.531</u>	<u>0.539</u>	0.828	<u>0.653</u>	<u>0.257</u>	<u>0.621</u>	0.648	0.374	<u>0.777</u>	0.777
MLP	0.090	0.705	0.731	0.742	0.736	0.642	0.743	0.749	<u>0.229</u>	<u>0.553</u>	0.857

Table 4. Results of the classification methos using Emotion dataset

	BR										
	HL↓	Acc↑	Prec↑	Rec↑	F_M↑	SA↑	MaFI↑	MiFI↑	OE↓	Cov↓	AP↑
kNN	0.188	0.551	0.687	0.641	0.663	0.307	0.678	0.653	0.256	1.775	0.806
DT	0.261	0.432	0.543	0.566	0.553	0.164	0.577	0.566	0.403	2.622	0.687
SVM	0.194	0.532	0.671	0.617	0.642	0.280	0.661	0.626	0.292	<u>2.401</u>	0.750
NB	0.225	0.549	0.618	0.755	0.679	0.248	0.675	0.655	0.307	1.851	0.789
MP	0.218	0.516	0.626	0.645	0.635	0.241	0.647	0.626	0.294	1.860	0.785
	LP										
kNN	0.215	0.560	0.649	0.671	0.659	0.336	0.661	0.647	0.365	2.319	0.395
DT	0.263	0.463	0.574	0.575	0.574	0.216	0.578	0.561	0.439	<u>2.608</u>	0.495
SVM	0.198	0.584	0.677	0.698	0.687	0.351	0.688	0.675	0.310	2.235	0.369
NB	0.229	0.519	0.640	0.635	0.637	0.263	0.630	0.616	0.393	2.345	0.438
MP	0.236	0.516	0.632	0.624	0.628	0.268	0.622	0.606	0.365	2.423	0.444
	RAKEL										
kNN	0.198	0.577	0.679	0.699	0.688	0.332	0.686	0.664	0.292	2.145	0.776
DT	0.234	0.502	0.612	0.631	0.621	0.238	0.630	0.619	0.326	1.986	0.766
SVM	0.186	0.592	0.710	0.703	0.706	0.341	0.701	0.681	0.260	1.989	0.798
NB	0.251	0.518	0.596	0.711	0.647	0.218	0.635	0.617	0.326	2.102	0.761
MP	0.209	0.556	0.670	0.677	0.673	0.295	0.666	0.650	0.289	1.964	0.787

previous dataset, the statistical test confirmed the predominance of the methods with the best results since these results are statistically significant to at least one other learning method for all 33 analyzed cases.

Table 4 shows the performance achieved when using the emotions dataset. For this dataset, once again, when using BR, k-NN presented the best results in most of the cases (8 out of 11). For LP, once again, SVM provided the best results in the majority of cases (10 out of 11), followed by DT (1 out of 11). In using RAKEL, it is seen that SVM presented the best results in most of the cases (9 out of 11), followed by DT and NB (1 out of 11). The results of the statistical test showed that although there is always a learning method with the best result, there is no statistical evidence to state that this result is significant different from the results provided by the other learning methods for the majority of the cases.

A general analysis of the performance of the learning methods, it can be observed that, as it was expected, SVM provided the highest number of best results (43 out of 99), followed by k-NN (34). The other learning methods had performance much poorer

than these two methods. When analyzing the best results for the different multi-label classification methods, k-NN provided the highest number of the best results for BR, while SVM had the best results for LP. For RAKEL, although SVM provided the highest number of the best results, it was closely followed by k-NN.

In the analysis of the best multi-label classification method, the best results of all three methods were compared and the top best result is selected. Based on this, it can be observed that RAKEL achieved the best performance (73 top best results), followed by LP (52) and BR (40), respectively. This was an expected result, since RAKEL and LP take into account the correlations between the labels. Moreover, RAKEL is an extension of LP, aiming to solve some of its problems, such as the large number of label subsets in which the majority of which are associated with very few examples.

6 Final Remarks

This paper presented a comparison between different methods of multi-label classification for different domain application, more specifically three different domains (datasets). In the experimental results, SVM usually presented the overall best results for all datasets. However, these best results are statistically significant for basically for scene dataset. This may have occurred because SVM manages better with features from the datasets and methods used. Of the multi-label methods, RAKEL has provided the best results. This was an expected result, since, in RAKEL and LP, the correlations between the labels are taken into account. Moreover, RAKEL is an extension of LP, aiming to solve some of its problems.

References

1. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data. In: Maimon, O., Rokach, L. (eds.) *Data Mining and Knowledge Discovery Handbook*, 2nd edn. Springer, Heidelberg (2010)
2. Cerri, R., Silva, R.R., Carvalho, A.C.: Comparing Methods for Multilabel Classification of Proteins Using Machine Learning Techniques. In: Guimaraes, K.S., Panchenko, A., Przytycka, T.M. (eds.) *Advances in Bioinformatics and Computational Biology*. LNCS, vol. 5676, pp. 109–120. Springer, Heidelberg (2009)
3. Zhang, M.L., Zhou, Z.H.: MI-knn: A lazy learning approach to multi-label learning. *Pattern Recognition* 40, 2038–2048 (2007)
4. Qi, G.J., Hua, X.S., Rui, Y., Tang, J., Mei, T., Zhang, H.J.: Correlative multi-label video annotation. In: 15th International Conference on Multimedia, New York, NY, USA (2007)
5. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In: Siebes, A., De Raedt, L. (eds.) *PKDD 2001*. LNCS (LNAI), vol. Principles of Data Mining and Knowledge Discovery, p. 42. Springer, Heidelberg (2001)
6. Zhang, M., Zhou, Z.: Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE T Knowl. & Data Eng.* 18(10), 1338–1351 (2006)
7. Tsoumakas, G., Friberg, R., Spyromitros-Xiou, E., Katakis, I., Vilcek, J.: Mulan software - java classes for ML classification, <http://mlkd.csdl.auth.gr/multilabel.html>
8. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: Proc. 9th Int. Conf. on Music Information Retrieval (2008)
9. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. *J. of Mach. Learn. Res.* 7, 1315–1338 (2007)
10. McCallum, A.: Multi-label text classification with a mixture model trained by em. In: Proc of AAAI 1999 Workshop on Text Learning (1999)

Learning Bimanual Coordination Patterns for Rhythmic Movements

Rikke Amilde Løvlid and Pinar Öztürk

Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway

Abstract. Coordinated bimanual movements form the basis for many everyday motor skills. In human bimanual coordination there are several basic principles or default coordination modes, such as the preference for in-phase or anti-phase movements. The objective of our work is to make robots learn bimanual coordination in a way that they can produce variations of the learned movements without further training. In this paper we study an artificial system that learns bimanual coordination patterns with various phase differences, frequency ratios and amplitudes. The results allow us to speculate that when the relationship between the two arms is easy to represent, the system is able to preserve this relationship when the speed of the movement changes.

1 Introduction

Humans use both hands in most of their daily tasks. Bimanual movements have, therefore, been rigorously studied in various research disciplines. There are mainly two dominant frameworks for the theory of interlimb coordination, the *dynamic pattern theory* [1,2] and *crosstalk theory* [3,4]. Dynamic pattern theory aims at a mathematical formalization of the coordination principles, modeling rhythmic movements as a system of coupled nonlinear oscillators [4]. The main idea in neural crosstalk theory is that interactions occur between command streams within a highly linked neural medium. These will give rise to patterns of mutual interference between concurrent limb motions at different stages of movement planning and organization [3].

In contrast to plentiful studies in psychology and neuroscience, robotic studies of bimanual coordination are surprisingly elusive. If robots are aimed to truly assist humans, they should be able to learn mastering movements that require bimanual coordination.

Our work is inspired by the hypothesis that the crosstalk-based bimanual coordination requires a peculiar connection between the motor system of the two limbs to be coordinated [5]. This paper presents an architecture that models the cross talk between two arms where the position of one arm is communicated to and controls the movement of the other arm, hence bimanual coordination. Our aim is not to investigate how the neural crosstalk happens in the nature but to develop an architecture that mimics the bimanual capabilities of humans.

The paper proposes a method to study the characteristics of the movements that the cross talk architecture can afford (i.e., learn and produce). Three properties are investigated to characterize and analyze the affordance of the architecture: *relative phase*, *frequency*, and *amplitude*. Then, the speed changes have been used as the evaluation criterion to evaluate the stability of the architecture with respect to the identified movement characteristics.

Our results indicate that the architecture acquires various movements through learning how to represent the interaction between the limbs, rather than each controller learning its movement in isolation.

2 The Method

The following three properties pertinent to a movement characterize the class of movement that the architecture is able to learn and produce.

Relative phase is the extent of phase lag between two joints at any point in time. Generally humans show a basic tendency towards perfectly synchronized, in-phase (phase difference $\Phi = 0^\circ$), or anti-phase movements ($\Phi = 180^\circ$) [6].

The *frequency* is the number of occurrences of a rhythmic, repeating movement per unit time. When doing rhythmic movements with both arms, humans have a tendency towards rhythms where the frequency of one arm is an integer multiple of the frequency of the other (e.g. 1:1, where both arms move together, 1:2, where one arm completes two cycles at the same amount of time as the other completes one, 1:3, etc.), as opposed to polyrhythms (e.g. 2:3 or 3:5) [4].

One limb moves with a larger *amplitude* than the other if it moves further than the other in a given time interval. For humans the amplitudes of the two arms have a tendency to become similar [4].

We elaborate the characteristics of movements in terms of these properties. Once a movement with one of these characteristics is learned, we test whether the cross talk architecture can preserve this characteristic with a new speed.

3 Architecture and Implementation

The bimanual coordination architecture relies on a simple crosstalk where the control system controls only the movement of the lagging arm; the movement of the leading arm is forced. The input to the system is the current states (the arms' joint angles), while the output is the motor command to the lagging arm.

The basic movement used in the experiments was simply raising and lowering the arms, with no movement in the elbow joint. The movement of each arm was described in two degrees of freedom, the angle between the arm and the body and the angle between the under- and upper arm. Hence, the simulated robot was described by 4 degrees of freedom.

The system architecture is shown in figure 1 and consists mainly of an inverse model. Generally an inverse model calculates an action that will bring the system to the desired next state. In the current setting, the desired next state is implicit in the state of the leading arm. The inverse model is acquired through

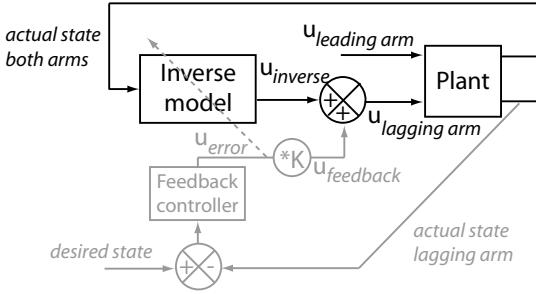


Fig. 1. The control architecture. An arrow represents training, and u is a motor command.

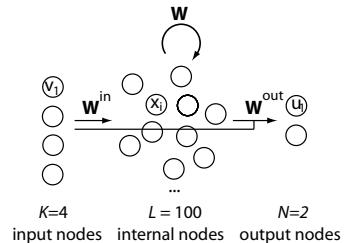


Fig. 2. The Inverse model as an Echo State Network

feedback-error-learning as suggested by Kawato, where a simple, analytical feedback controller is used together with the inverse model [7].

The inverse model was implemented as an *echo state network* (ESN) [8] as illustrated in figure 2. The input to the inverse model, v , is the current state from the plant (the robot simulator);

$$\mathbf{v}^{t+1} = \text{plant}(\mathbf{u}_{\text{inverse}}^t + K\mathbf{u}_{\text{feedback}}^t, \mathbf{u}_{\text{leading arm}}^t). \quad (1)$$

The feedback gain K , which decides how much the feedback controller is able to influence the final motor command, was linearly decreased from 1 to 0 during training. The activation of the internal nodes was updated according to

$$\mathbf{x}^{t+1} = f(\mathbf{W}^{\text{in}} \mathbf{v}^{t+1} + \mathbf{W} \mathbf{x}^t), \quad (2)$$

where f is the activation function. The motor command calculated by the inverse model is given by

$$\mathbf{u}_{\text{inverse}}^{t+1} = f^{\text{out}}(\mathbf{W}^{\text{out}}(\mathbf{v}^{t+1}, \mathbf{x}^{t+1})). \quad (3)$$

The training procedure was organized in epochs and cycles, where one cycle is one full temporal presentation of the training motion. Each epoch consisted of seven cycles. First, the network was re-initialized by setting the internal states of the network to zero, and one cycle was run without updating the weights. Subsequently, the training sequence was presented five times with enabled learning. The output connections were then adapted after each complete cycle. A final cycle was used to estimate the performance error on the training sequence while learning was disabled.

During the training cycles, the state of each node was stored in a state collection matrix, \mathbf{M} , and $(f^{\text{out}})^{-1}(\mathbf{u}_{\text{target}})$ was collected row-wise into a target collection matrix, \mathbf{T} . Equation 3 can now be written

$$\mathbf{M}(\mathbf{W}^{\text{out}})^T = \mathbf{T}, \quad (4)$$

and solved with regard to \mathbf{W}^{out} with the use of the Moore-Penrose pseudoinverse:

$$(\mathbf{W}^{\text{out}})^T = \mathbf{M}^+ \mathbf{T}. \quad (5)$$

Note that it is not straight forward to decide how to compute the target motor command, \mathbf{u}_{target} , used in the target collection matrix. The desired motor command is only known indirectly through the state of the leading arm. Generally, several motor commands may result in the same position of the limbs, and one does not want to bias the controller into choosing one specific solution. Because we had batch learning, the best guess could be calculated by using the u_{error} provided in the next time step, $u_{best\ guess}^t = u^t + u_{error}^{t+1}$ (e_{error}^t reflects the error done at time step $t - 1$). However, using $u_{best\ guess}$ as the target did not lead to the best solution. In previous work we have shown that using something in between the best guess and the actual output of the inverse model in the previous cycle yields better results [9]. This result was confirmed in the current experiments. The target state for time step t in epoch $k + 1$ was then given by:

$$u_{target}^{t,k+1} = \beta u_{best\ guess}^t + (1 - \beta) u_{inverse}^{t,k} \quad (6)$$

In the experiments we used $\beta = 0.01$.

After the inverse model was trained on one movement, we wanted to test how the changes in the movement of the leading arm would affect the movement of the lagging arm. This was done by changing the speed of the leading arm and run the network for one additional epoch with only two cycles, the initialization cycle and the evaluation cycle. The feedback gain K was 0 during testing.

4 Experiments and Results

In the experiments the control system was trained to learn a coordinated movement. The stability of different coordination patterns was compared by testing the system's ability to replicate the learned pattern in different speeds.

Relative Phase: In this experiment the arms were trained to move in a specific phase relationship. It could be in-phase ($\Phi = 0^\circ$), i.e., both arms are raised and lowered synchronously, or anti-phase ($\Phi = 180^\circ$), i.e., one arm is lowered as the other is raised, or anything in between. During the training, the control system learned to perform one specific phase relationship for one motion in one speed. During testing, the speed of the leading arm was changed.

The control system had no problem generalizing to new speeds when it was trained (and tested) with in-phase or anti-phase movements, as illustrated in figure 3(a).

The results of testing the system in higher or slower speeds than it was trained for when the two arms are in 90° phase difference is shown in figure 3(b). The control system is trying to coordinate its movement according to the movement of the leading arm, instead of sticking to the absolute timing it was trained on, but it does not accomplish this as well as it did in the in- and anti-phase modi. The control system has a hard time matching the speed of the leading arm. When the leading arm moves faster, the lagging arm also moves faster, but not as much as the leading arm. Equivalently, when the speed of the leading arm is decreased, the lagging arm also moves slower, but not slow enough. Instead of changing

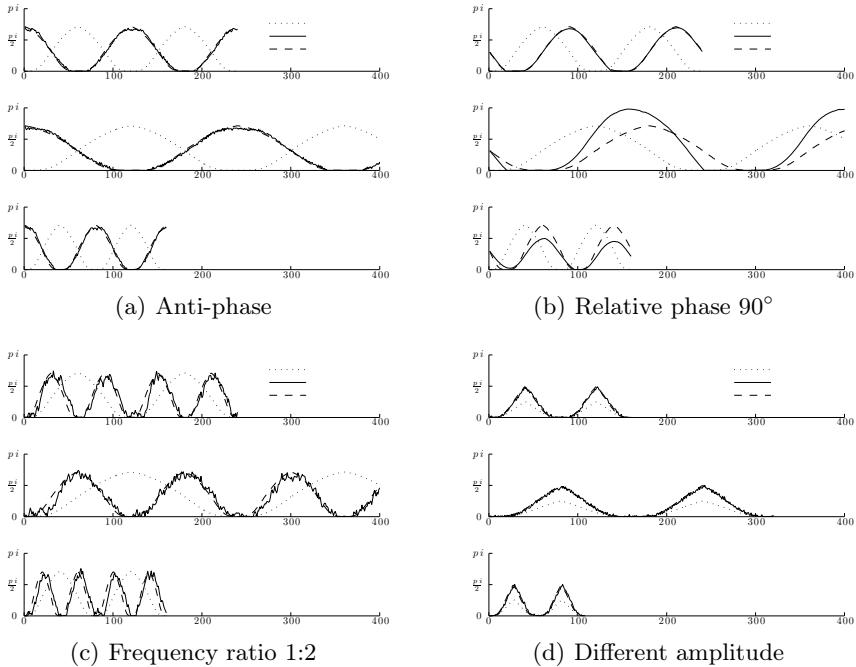


Fig. 3. Results when the trained control system was tested on the same movement in the original speed (top), half the original speed (middle) and on 50% increase of the original speed (bottom). The shoulder angle of the leading arm and the desired and actual shoulder angle of the lagging arm at each time step is plotted. The movement of the leading arm is shown together with the desired and actual movement of the controlled lagging arm. (a) The generalization to different speeds is close to perfect for the anti-phase movement. (b) The phase difference of 90° seems harder to preserve when speed is changed. (c) The system was trained to perform the basic movement with frequency ratio 1:2. The movement of the lagging arm is not smooth, but the timing is just as good in the testing case as in the training case. (d) The lagging arm moves twice as far within the same time interval as the leading arm. The control system has no problem generalizing.

direction at the trained state, the leading arm overshoots or undershoots the target when moving too fast or too slow respectively. As a consequence, the timing of the change in direction is closer to the desired than what it would have been if the change in direction had been timed solely according to the state of the lagging arm.

Frequency: The objective in this experiment was to study the control system's capability to perform a different rhythm with each arm. The control system trained both with simple rhythms, with frequency ratio 1:2 (leading arm moves ones while lagging moves twice) and 1:3, and more complex polyrhythms with frequency ratio 2:3 and 3:5.

This proved to be quite difficult. Training with frequency ratio 2:3 failed, and as illustrated in figure 3(c) (top), the performance is far from smooth for frequency 1:2. However, the timing for frequency 1:2 is satisfactory, and the results when testing the network trained on novel speeds is not worse than when tested on the original speed, as illustrated in figure 3(c) (middle and bottom).

Amplitude: In this experiment the arms were trained to move with different amplitude with respect to each other, one arm making a larger movement than the other, i.e. the motor commands of one of the arms must be twice as large as the other's. This appeared to be an easy task to learn, and the control system generalized perfectly to novel speeds as illustrated in figure 3(d).

5 Discussion and Conclusion

The aim of the work was to develop an artificial system that could learn and generalize different bimanual coordination patterns related to relative phase, frequency ratio and amplitude in the proposed architecture. As it is with humans, in-phase and anti-phase movements was more stable than the movements with other phase relationships. As for frequency ratio, polyrhythms like 2:3, appeared to be impossible to learn, whereas simpler rhythms like 1:2 and 1:3 seemed relatively stable once learned. The control system had no problem generalizing when it was trained with different amplitudes on the two arms. These results allow us to speculate that when the relationship between the two movement components is easy to represent, the system is able to preserve this relationship when the speed of the movement changes. In the continuation of this work we will try to more formally define the characteristics of the movement that are easier to coordinate, and what is easy to represent.

References

1. Kelso, J.A.S.: Dynamic patterns: The self-organization of brain and behavior (1995)
2. Kelso, J.A.S., de Guzman, G.C., Reveley, C., Tognoli, E.: Virtual partner interaction (vpi): exploring novel behaviors via coordination dynamics. *PLoS One* 4(6) (2009)
3. Banerjee, A., Jirsa, V.K.: How do neural connectivity and time delays influence bimanual coordination? *Biological Cybernetics* 96(2), 265–278 (2007)
4. Swinnen, S.P.: Intermanual coordination: From behavioural principles to neural-network interactions. *Nature* 3, 348–359 (2002)
5. Diedrichsen, J., Shadmehr, R., Ivry, R.B.: The coordination of movement: optimal feedback control and beyond. *Trends in Cognitive Sciences* (in press)
6. Peper, C.E., Beek, P.J.: Modeling rhythmic interlimb coordination: The roles of movement amplitude and time delays. *Human Movement Science* 18, 263–280 (1999)
7. Kawato, M., Gomi, H.: The cerebellum and vor/okr learning models. *TINS* 15(11) (1992)
8. Jaeger, H.: A tutorial on training recurrent neural networks, covering bppt, rtrl, and the echo state network approach. *Techreport, Fraunhofer Institute for AIS* (2008)
9. Løvlid, R.A.: Introducing learning rate analogy to the training of echo state networks. In: *First Norwegian Artificial Intelligence Symposium* (2009)

Classification of Voice Aging Using Parameters Extracted from the Glottal Signal

Leonardo Alfredo Forero Mendoza¹, Edson Cataldo², Marley Vellasco¹,
and Marco Silva¹

¹ Eletrical Engineering Department - Pontifical Catholic University of Rio de Janeiro
CEP 22.451-900 Rua Marquês de São Vicente, 225 Gávea - Rio de Janeiro, Brasil
{Mendonza, Marley, mabs21}@ele.puc-rio.br

² Applied Mathematics Department –Telecommunications Engineering -
Universidade Federal Fluminense
CEP 24210-240 Escola de Engenharia - Bloco D - Sala 502-B R. Passo da Pátria, 156,
São Domingos Niterói, RJ Brasil
ecataldo@im.uff.br

Abstract. Classification of voice aging has many applications in health and geriatrics. This work focuses on finding the most significant parameters to identify voice aging. This work proposes to choose the most significant parameters extracted of the glottal signal to identify the voice aging process of men and women using the wrapper approach combining a genetic algorithm (as a search algorithm) with a neural network (as an induction algorithm). The chosen parameters will be used as entries in a neural network to classify male and female Brazilian speakers in three different age groups, which will be called young (from 15 to 30 years old), adult (from 31 to 60 years old) and senior (from 61 to 90 years old). The voice database used for this work was composed by one hundred twenty Brazilian people (male and female) of different ages. In this work we use the largest basis for classification of age compared with other similar works, and its rate of classification is superior to other studies reaching 91.6% in males and 83.33% in women.

Keywords: voice aging, wrapper approach, glottal flow, glottal signal.

1 Introduction

From 1995 to 2005, the life expectancy of Brazilian people has increased more than 8%. This scenario is a global tendency, resulting in the need for developing new treatments to provide better life quality to senior people, including care for their voice [1]. The most common method for extracting voice features is directly from the voice signal. However, many researchers have looked for some characteristics extracted from the so called glottal signal, which is the signal obtained just after the vocal folds and before the vocal tract. Since one of the consequences of aging is the change of the vocal fold structures, this signal is, therefore, important because it preserves characteristics from the movement of the vocal folds without the influence of the vocal tract. Nowadays, obtaining this signal is easier due to the development of algorithms that

can perform an inverse filtering from the voice signal, eliminating the influence of the vocal tract. Before, it used to be necessary to use equipment coupled with micro cameras to record sounds just after air passed from the vocal folds. This was an invasive technique and very difficult to be performed. In previous work has been used to classify Bayes, Hidden, Markov Models (HMM), Gaussian Mixture Models (GMM) and neural networks, using as input Mel-frequency cepstral Coefficients (MFCC) and the parameters jitter and shimmer, adding 37 parameters entry [3] [4], using bases of 12 recordings for men and women not getting the results that rise from 80% [3]. The MFCC have proven effectiveness for speaker recognition [6], but their performance is not equal to the classification of voice aging. In [4] compare several classifiers for classification of voice aging, getting the best results with the techniques of neural networks and support vector machine, the difference in the recognition of speaker that gets the best results with GMM and HMM. This is because for classification of voice aging are not as important temporal features of the voice and to change the voice signal with age [4]. The great advantage of this work is the number of voice used and the classification is used only the most relevant parameters found by the wrapper method which reduces the input parameters of the neural network.

2 Features Extraction from the Glottal Signal

2.1 The Glottal Signal

The voice signal, particularly the one related to voiced sounds, e.g. vowels, starts with the contraction-expansion of the lungs, generating a pressure difference between the air in the lungs and the air near the mouth. The airflow created passes through the vocal folds, which oscillate in a frequency called the fundamental frequency of the voice. The pressure signal formed by the air pulses is quasi-periodic and it is called the glottal signal [12]. Nowadays, it is possible to obtain the glottal signal using non invasive methods, by performing an inverse filtering on the voice signal, which consists in eliminating the influence of the vocal tract and the voice radiation caused by the mouth, preserving the glottal signal characteristics [6]. In this paper, the inverse filtering algorithm used is of the semi-automatic category, called PSIAIF (Pitch Synchronous Iterative Adaptive Inverse Filtering) [7] [8]. There is a toolbox implementation in Matlab, called Aparat [9], which was constructed especially based on PSIAIF method to obtain the glottal signal as well as the extraction of its main features or parameters.

2.2.1 The Time Domain Parameters

The time domain parameters that can be extracted from the glottal signal are described below [7], [8].

- (i) Closing phase (K_o): it describes the interval between the instant of the maximum opening of the vocal folds and the instant where they close [7].
- (ii) Opening phase (K_a): it describes the interval between the instant where the vocal folds start the oscillation up to their maximum opening [7].
- (iii) Opening quotient (OQ): The ratio between the total time of the vocal folds opening and the total time of a cycle (or period) of the glottal signal (T). It is inversely proportional to the intensity of the voice. [1][7].

- (iv) Closing quotient (CIQ): The ratio between the closing phase parameter (Ko) and the total length of a glottal pulse (T) [7]. It is inversely proportional to voice intensity. The smaller it is, the higher the voice intensity.
- (v) Amplitude quotient (AQ): The ratio between the glottal signal amplitude (Av) and the minimum value of the glottal signal derivative ($dmin$) [10]. It is related to the speaker phonation [8].
- (vi) Normalized amplitude quotient (NAQ): It is calculated by the ratio between the amplitude quotient (AQ) and the total time length of the glottal pulse (T) [11].
- (vii) Opening quotient defined by the Liljencrants-Fant model (OQa): This is another opening quotient but calculated by the Liljencrants-Fant model for inverse filtering. Details about this model can be found in [12].
- (viii) Quasi opening quotient (QoQ): It is a relationship between the glottal signal opening at the exact instant of the oscillation and the closing time [9]. It is used in some research to classify emotions [13].
- (ix) Speed quotient (SQ) is defined as the ratio of the opening phase length to the closing phase length [7].

2.2.2 The Frequency Domain Parameters

- (i) Difference between harmonics (DH12): Also known as H1-H2 and it is the difference between the values of the first and second harmonics of the glottal signal [14][15]. This parameter has been used to measure vocal quality.
- (ii) Harmonics relation factor (HRF): It relates the first harmonic ($H1$) with the sum of the energy of the other harmonics (Hk) [16]. It has also been used to measure the vocal quality.
- (iii) Jitter: variations in fundamental frequency between successive vibratory cycles [17] [18]. Changes in jitter may be indicative of neurological or psychological difficulties [1].
- (iv) Shimmer: variations in amplitude of the glottal flow between successive vibratory cycles [17] [18]. Changing the shimmer is found mainly in the presence of mass lesions in the vocal folds, such as polyps, edema or carcinomas [1].
- (v) Harmonics-to-Noise-Ratio (HNR): is an acoustic measure that determines how much of harmonics (periodic) and of noise (aperiodic) exists in a voice. This measurement has proved to be a sensitive and strong indicator of vocal productions in adults [1].

3 Model Used for Voice Aging Classification

The model has three stages, the first stage is to obtain the parameters of the signal through the glottal signal, the second stage is to choose the best parameters to classify aging voice with wrapper feature selection and the third stage is classification of aging by voice using a multilayer perceptron type neural network with the parameters chosen in the second stage.

3.1 Inverse Filtering

For each vocal register the corresponding glottal signal was obtained by inverse filtering PSIAIF and the parameters were extracted using the Aparat [10] and praat [23] software. The following parameters were obtained: fundamental frequency (fo), jitter, shimmer, HNR, Ko , PP, NAQ, AQ, CIQ, OQ1, OQ2, Oqa, Qog, SQ1, SQ2, DH12 and HRF. The parameters were separated according to the groups to which they belonged. OQ was divided into $OQ1$, $OQ2$, the open quotients calculated from the so-called primary and secondary openings of the glottal flow; SQ was divided into speed quotients calculated from the primary and the secondary openings of glottal signal.

3.2 Wrapper Feature Selection

A parameter selection method aims to find a subset of the most relevant attributes/parameters, for a specific task, from the original set of attributes. It intends to make the learning process more efficient, since redundant attributes can reduce the classification performance in terms of computational time (due to the large quantity of data) in terms of generalization capacity [19]. To find the relevant subset the wrapper approach combines a search algorithm herein a genetic algorithm, with an induction algorithm, herein an artificial neural network. The genetic algorithm optimizes the results of the induction algorithm choosing the most relevant subsets for the classification. The genetic algorithms are search and optimization methods inspired by the natural evolution process and by the natural selection proposed by Charles Darwin [17]. To choose the most relevant parameters using the wrapper approach, genetic algorithm binary was applied with the following characteristics: One-point crossover of 0.6, mutation of 0.03 and population of 40 individuals by 60 generations using the 17 parameters as the chromosome representation, each gene that makes up the chromosome represents a parameter of the glottal signal. The fitness function used was the accuracy of the classifier; in this case the neural network type Multilayer Perceptron (MLP). The neural network input is each chromosome resulting after passing through the genetic algorithm operators for which the number of entries is not fixed. The hidden layer has three neurons and the network is trained with 500 epochs of training. After applying the genetic algorithm, different parameters were considered as most relevant for the aging identification for female and male. The subset containing shimmer and AQ was defined for the female group and the subset consisting of Ko , DH12, NAQ, OQ1, OQ2, Oqa and SQ2 for the male group. Then, a study was performed concerning these parameters in order to identify why they had been chosen.

3.3 Classification of Voice Aging Using Neural Networks

With the parameters of the glottal signal selected by the wrapper, are classified with a neural network (MLP). The neural network classification is developed to classify the speaker into the three groups. These groups will be the outputs of the neural network in which 65% from the database were used for training, 15% as validation set (to avoid the network overtraining and to choose the number of processors and the hidden layers) and 20% for testing.

4 Database

The database used for this work is composed of 60 male voice registers and 60 female voice registers for the sustained vowel /e/ for the Portuguese spoken in the city of Rio de Janeiro city, Brazil. This database was divided in three different age groups: from 15 to 30 years old, from 31 to 60 years old and from 61 to 90 years old. There were 20 voice registers for each age group.

5 Results

5.1 Classification of Voice Aging Using Neural Networks

For the female database, the best result was achieved using 3 neurons with only one hidden layer classifying correctly 10 speakers and incorrectly 2 speakers; that is, 83.33% for guess rate. For the male database, the best result was achieved using 4 neurons with only one hidden layer classifying correctly 11 speakers and incorrectly 1 speaker; that is, 91,66% for guess rate.

5.2 Analyses of Parameters Chosen for Classification

In the case of the female database, the shimmer has already been identified as a parameter which greatly increases in different periods of the women's life [1], [20]. The mean values for the shimmer parameter were calculated and it was observed that the minimum value was obtained for the group 15-30 years old.

The box plot, constructed for the Shimmer parameter is shown in the Fig.1. The AQ parameter has a similar behavior as the shimmer parameter. Its box plot is shown in Fig. 2. The box plot draws the median value.

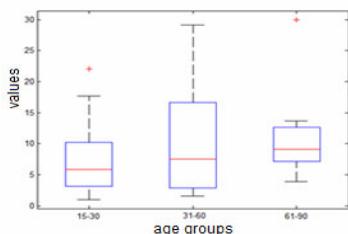


Fig. 1. Box plot shimmer for female

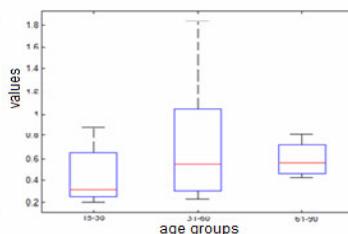


Fig. 2. Box plot AQ for female

As can be noticed, the Shimmer parameter does not present a significant variation after 30 years old, which can prevent the neural network classification model from obtaining better performance in differentiating the 31-60 and 61-90 groups. Shimmer and AQ has mean value and median very similar to the range of 31-60 and 61-90, but the variance is lower in the range of 61-90, that explains why women after menopause 45-55 years or so has a sudden change in hormones produce changes in voice. In the

male database, the parameters chosen were the subset composed of Ko , $DH12$, NAQ , $OQ1$, $OQ2$, Oqa and $SQ2$. Parameters $OQ1$, $OQ2$, Oqa and NAQ are related to the voice intensity and all of them have the same behavior. In this specific group, the mean values for the 61-90 group are very high when compared to other groups, as shown in figures. 3 to 9. The mean value for Ko for male group is high in the 61-90 group when compared to other groups. It may occur due to the vocal folds relax, during the aging process according to the aging, because the closing of the vocal folds takes more time. The mean values for the $SQ2$ parameter are higher for the 15-30 group. This parameter is related to the voice quality, as can be observed the voice quality in the 61-90 group normally decreases. The $DH12$ parameter does not show a large difference among the speakers.

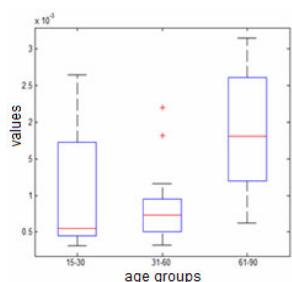


Fig. 3. Ko for male

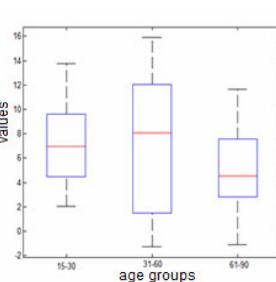


Fig. 4. $DH12$ for male

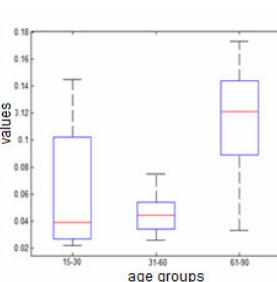


Fig. 5. NAQ for male

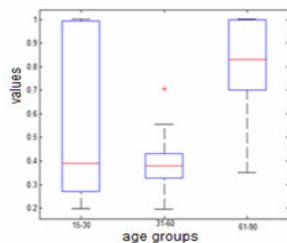


Fig. 6. $OQ1$ for male

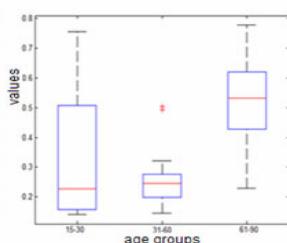


Fig. 7. $OQ2$ for male

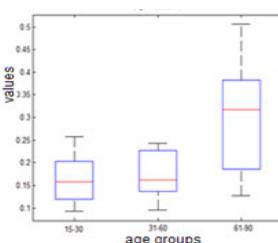


Fig. 8. OQa for male

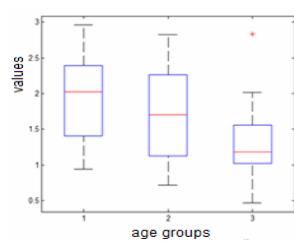


Fig. 9. $SQ2$ for male

It should be also noted that the parameters used to relate voice intensity and vocal quality, as OQ1 OQ2 and NAQ are good for identifying voice aging. In previous works in similar database [6] a 50% accuracy was obtained in the out of sample dataset, using only jitter and shimmer parameters. The use of other parameters whit Ko, SQ usually applied to identify pathologies and emotions, related to the voice, improved the aging classification rate [13], [16], [18].

6 Conclusions

This paper presents a model for choosing the most significant parameters of glottal signal to sort through aging voice of neural networks. The results obtained were satisfactory and better than those achieved in other works. However, it is important to note that it is not conclusive that separately the parameters will correctly identify the voice aging, but it is conclusive that when they are combined the performance is improved. It should be noted that the wrapper approach showed to be effective because the chosen subsets seemed to be strong discriminates of different speaker ages. This paper uses parameters from the glottal signal, which is a novelty.

Acknowledgments

The authors acknowledge the financial support from the Brazilian agencies CNPq (Conselho Nacional de Pesquisa e Desenvolvimento), CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and FAPERJ (Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro)

References

1. Rosa, I.S.: Analise acústica da voz de indivíduos na terceira idade. Tese de mestrado Universidade de São Carlos (2005)
2. Verdonck, I., Mahieu, H.: Vocal aging and the impact on daily life: a longitudinal study (2003), doi:10.1016/j.jvoice
3. Sadeghi, N.A., Homayounpour, M.M.: Speaker age interval and sex identification based on jitters, shimmers and mean MFCC using supervised and unsupervised discriminative classification methods. In: Proc. ICSP, Guilin, China (2006)
4. Sedaaghi, M.H.: A Comparative Study of Gender and Age Classification in Speech Signals. Iranian Journal of Electrical & Electronic Engineering 5(1) (March 2009)
5. Campbell Jr., J.P.: Speaker Recognition: A tutorial. Proceedings of the IEEE 85(9), 1437–1462 (1997)
6. Alku, P.: Glottal wave analysis with Pitch Synchronous Adaptive Inverse Filtering”. Speech Communication 11, 109–118 (1992)
7. Pulakka, H.: Analysis of Human Voice Production Using Inverse Filtering, High-Speed Imaging, and Electroglossography. University of Technology Helsinki (2005)
8. Juliano, S. M.: Um estudo comparativo entre o sinal electroglotográfico e o sinal de voz”, Dissertação de mestrado em Engenharia de Telecomunicações, UFF (2008)

9. Software Aparat Helsinki University of Technology Laboratory of Acoustics and Audio Signal Processing,
http://aparat.sourceforge.net/index.php/Main_Page
10. Alku, P., Vilkman, E.: Amplitude Domain Quotient for Characterization of the Glottal Volume Velocity Waveform Estimated by Inverse Filtering. *Speech Communication* 18(2), 131–138 (1996)
11. Alku, P., Bäckström, T., Vilkman, E.: Normalized Amplitude Quotient for Parameterization of the Glottal Flow. *Journal of the Acoustical Society of America* 112(2), 701–710 (2002)
12. Gobl, C., Chasaide, A.: Amplitude-based source parameters for measuring voice quality. In: VOQUAL, pp. 151–156 (2003)
13. Laukkanen, A.-M., Vilkman, E., Alku, P.: Related to Stress and Emotional State: a Preliminary Study. *Journal of Phonetics* 24(3), 313–335 (1996)
14. Titze, I., Sundberg, J.: Vocal intensity in speakers and singers. *Journal of the Acoustical Society of America*, 2936–2946 (May 1992)
15. Airas, M.: Methods and studies of laringeal voice quality analysis in speech production. Dissertation for the degree of Doctor Helsinki University of Technology (2008)
16. Childers, D.G., Lee, C. K.: Vocal quality factors: Analysis, synthesis, and perception. *Journal of the Acoustical Society of America*, 2394–2410 (May 1990)
17. Cataldo, E., Rodrigues, F., Brandão, A., Lucero, J.: Usando Redes Neurais para Classificação de padrões de voz. In: XXVIII CNMAC - Congresso Nacional de Matemática Aplicada e Computacional, 2005, Anais do XXVIII CNMAC, São Paulo (2005)
18. Vieira, M.N.: Automated Measures of Dysphonias and the Phonatory Effects of Asymmetries in the Posterior Larynx, Ph.D Thesis, University of Edinburgh, UK (1997)
19. Pappa, G.L.: Seleção de atributos utilizando algoritmos genéticos múltiplos objetivos. Tese de mestrado PUC Paraná (2005)
20. Behlau, M., Pontes, P.P.: Avaliação e tratamento das disfonias. Lovise, São Paulo (1995)
21. Software praat University of Amsterdam, <http://www.fon.hum.uva.nl/praat/>

TopoART: A Topology Learning Hierarchical ART Network

Marko Tscherepanow

Bielefeld University, Applied Informatics
Universitätsstraße 25, 33615 Bielefeld, Germany
marko@techfak.uni-bielefeld.de

Abstract. In this paper, a novel unsupervised neural network combining elements from Adaptive Resonance Theory and topology learning neural networks, in particular the Self-Organising Incremental Neural Network, is introduced. It enables stable on-line clustering of stationary and non-stationary input data. In addition, two representations reflecting different levels of detail are learnt simultaneously. Furthermore, the network is designed in such a way that its sensitivity to noise is diminished, which renders it suitable for the application to real-world problems.

Keywords: on-line learning, topology learning, Adaptive Resonance Theory, hierarchical clustering.

1 Introduction

For numerous tasks, the traditional off-line learning approach with separate training, validation, and test phases is not sufficient. The diagnosis of genetic abnormalities [1], interactive teaching of a humanoid robot [2], and the subcellular localisation of proteins [3] constitute some examples for such problems. As a consequence, incremental on-line learning has become more popular in recent years, since such machine learning techniques are required to gradually complete knowledge or adapt to non-stationary input distributions.

In this paper, a novel neural network is introduced which combines incremental and fast on-line clustering with topology learning. Based on its origins in Adaptive Resonance Theory (ART) networks, it is called TopoART. Thanks to these origins, TopoART creates stable representations while retaining its ability to learn new data. In order to render TopoART more suitable for real-world applications, it was designed in such a way that it becomes insensitive to noise. Furthermore, TopoART creates a hierarchical representation of the input distribution reflecting different levels of detail.

In Sect. 2, related learning methods are outlined. Afterwards, details of the TopoART algorithm are introduced in Sect. 3. Then, clustering results for artificial and real-world data are presented in Sect. 4. Here, the ability of TopoART to cope with noise and to incrementally learn new input data from non-stationary distributions will be demonstrated as well. Finally, Sect. 5 summarises the most important points of this paper.

2 Related Work

The k-means algorithm [4], which constitutes a very well-known unsupervised learning technique, determines a partitioning of an input distribution into k regions or rather clusters. Each cluster is represented by a reference vector. The determination of the number of required clusters constitutes a crucial problem. For this reason, the Linde-Buzo-Gray (LBG) algorithm [5] was developed. Based on a fixed training set, it successively computes sets of reference vectors of increasing size until a stopping criterion is fulfilled. The topological structure of the input data is not considered by this algorithm.

In 1982, the Self-Organising Feature Maps (SOFMs) [6], which map input data to a lattice of neurons, were introduced. Here, the reference vectors are encoded by the weights of the neurons. The lattice possesses a predefined topological structure, the dimension of which is usually lower or equal to the dimension of the input space. If the input distribution is not known in advance, an appropriate lattice structure is very difficult to choose. This problem was solved by the Growing Neural Gas (GNG) algorithm [7]. It allows for the incremental incorporation of new neurons and the learning of the input distribution's topology by adding and deleting edges between different neurons. But the representation of the input distribution is not stable: A continuing presentation of input data, even if they have already been learnt, results in a continuing adaptation of the neurons' weights, i.e. the reference vectors, and the network topology. As a consequence, already acquired knowledge gets lost due to further training. This problem has been termed the *stability-plasticity dilemma* [8]. It occurs, for instance, if the input distribution is complex. But small changes of the input probabilities or the sequencing of the input data may cause a similar effect.

Adaptive Resonance Theory (ART) networks have been proposed as a solution to the stability-plasticity dilemma [8]. These networks learn top-down expectations which are matched with bottom-up input. The expectations, which are called categories, summarise sets of input data to clusters. Depending on the type of ART network, the categories exhibit different shapes such as a hyper-spherical shape [9], an elliptical shape [10], or a rectangular shape [11]. Besides enabling ART networks to create stable and plastic representations, the categories allow for an easy novelty detection. But in contrast to SOFMs and GNG, ART networks do not capture the topology of the input data. Furthermore, their ability of stable learning leads to an increased sensitivity to noise.

In 2006, the Self-Organising Incremental Neural Network (SOINN) was introduced [12]. Similar to GNG, SOINN clusters input data by incrementally adding neurons, the weights of which represent reference vectors, and representing the topology by edges between nodes. But it has several additional features: Firstly, SOINN has a two-layered structure representing the input distribution at different levels of detail. Additionally, this structure decreases the sensitivity to noise. The second layer is trained after the training of the first layer has been finished. Secondly, novelty detection can be performed based on an adaptive threshold. Thirdly, each neuron has an individual learning rate which decays if the amount of input samples it represents increases. By this means, a more stable

representation is achieved. But the weights of the neurons do not stabilise completely. Furthermore, a high number of relevant parameters (8 parameters per layer) has to be selected in order to apply SOINN.

The Enhanced Self-Organising Incremental Neural Network (ESOINN) [13] solves some of the above mentioned problems: By removing the second layer and one condition for the insertion of new neurons, the number of required parameters is considerably decreased (4 in total). Furthermore, the whole network can be trained on-line. But similar to SOINN, the weights do not stabilise completely. Moreover, ESOINN loses the ability to create hierarchical representations.

TopoART combines the advantages of ART and topology learning networks. From its ART ancestors, it inherits the ability of fast and stable on-line learning using expectations (categories). But the categories are extended by edges reflecting the topology of the input distribution enabling the formation of arbitrarily shaped clusters. In addition, it adopts the ability to represent input data at different levels of detail from SOINN; but unlike SOINN, TopoART learns both levels simultaneously.

3 The TopoART Algorithm

The basic structure and the computational framework of TopoART are strongly related to Fuzzy Art [11] – a very efficient ART network utilising rectangular categories. Fuzzy ART constitutes a three-layered neural network (see Fig. 1). Input is presented to the initial layer $F0$. Here, the input vectors $\underline{x}(t)$ are encoded using complement coding. The encoded input vectors are denoted by $\underline{x}^{F1}(t)$. As a consequence of the usage of complement coding, each component $x_i(t)$ of an input vector $\underline{x}(t)$ has to lie in the interval $[0, 1]$.

$$\underline{x}(t) = [x_1(t), \dots, x_d(t)]^T \quad (1)$$

$$\underline{x}^{F1}(t) = [x_1(t), \dots, x_d(t), 1 - x_1(t), \dots, 1 - x_d(t)]^T \quad (2)$$

The encoded input vectors $\underline{x}^{F1}(t)$ are transmitted to the comparison layer $F1$. From here, they activate the neurons of the representation layer $F2$:

$$z_i^{F2}(t) = \frac{|\underline{x}^{F1}(t) \wedge \underline{w}_i^{F2}(t)|_1}{\alpha + |\underline{w}_i^{F2}(t)|_1} \quad (3)$$

The activation $z_i^{F2}(t)$ (choice function) constitutes a measure for the similarity between $\underline{x}^{F1}(t)$ and the category represented by neuron i . $|\cdot|_1$ and \wedge denote the city block norm and a component-wise minimum operation, respectively. The parameter α must be set slightly higher than zero. The choice of the actual value is not crucial.¹ In general, $z_i^{F2}(t)$ prefers small categories to large ones.

After all $F2$ neurons have been activated, the best-matching neuron bm , i.e. the neuron with the highest activation, is selected. But the category represented

¹ α was set to 0.001 for all experiments presented in this paper.

by its weights $\underline{w}_{bm}^{F2}(t)$ is only allowed to grow and enclose a presented input vector if resonance occurs, i.e. if the match function (4) is fulfilled.

$$\frac{|\underline{x}^{F1}(t) \wedge \underline{w}_{bm}^{F2}(t)|_1}{|\underline{x}^{F1}(t)|_1} \geq \rho \quad (4)$$

In case of resonance, the weights \underline{w}_{bm}^{F2} of the chosen neuron are adapted and the output $y(t)$ of the network is set:

$$\underline{w}_{bm}^{F2}(t+1) = \beta(\underline{x}^{F1}(t) \wedge \underline{w}_{bm}^{F2}(t)) + (1 - \beta)\underline{w}_{bm}^{F2}(t) \quad (5)$$

$$y_i(t) = \begin{cases} 0 & \text{if } i \neq bm \\ 1 & \text{if } i = bm \end{cases} \quad (6)$$

β denotes the learning rate. Setting β to 1, trains the network in fast-learning mode; i.e., each learnt input is enclosed by the category that matches it best. As the categories cannot shrink according to (5), the formed representations are stable. The current size $S_i(t)$ of a category can be derived from the weights $\underline{w}_i^{F2}(t)$ of the corresponding neuron i :

$$S_i(t) = \sum_{j=1}^d |(1 - w_{i,d+j}^{F2}(t)) - w_{i,j}^{F2}(t)| \quad (7)$$

The growth of the categories is limited by the vigilance parameter ρ and the dimension of the input space d , which determine the maximum category size S^{\max} .

$$S^{\max} = d(1 - \rho) \quad (8)$$

Assuming a neuron was not able to fulfil (4), its activation is reset. Then a new best-matching node is chosen. If no suitable best-matching neuron is found, a new neuron representing $\underline{x}(t)$ is incorporated and resonance occurs.

TopoART is composed of two Fuzzy ART-like components (TopoART *a* and TopoART *b*) using a common *F0* layer performing complement coding (see Fig. II). But in contrast to Fuzzy ART, each *F2* neuron i of both components has a counter denoted by n_i^a and n_i^b , respectively, which counts the number of input samples it has learnt. An encoded input vector is only propagated to TopoART *b* if resonance of TopoART *a* occurred and $n_{bm}^a \geq \phi$. Every τ learning cycles, all neurons with a counter smaller than ϕ are removed. Therefore, such neurons are called node candidates. Once n_i equals or surpasses ϕ , the corresponding neuron cannot be removed anymore; i.e., it becomes a permanent node. By means of this procedure, the network becomes insensitive to noise but is still able to learn stable representations.

In order to allow for fast on-line learning, the learning rate β_{bm} used for adapting the weights of the best-matching neuron is always set to 1 yielding:

$$\underline{w}_{bm}^{F2}(t+1) = \underline{x}^{F1}(t) \wedge \underline{w}_{bm}^{F2}(t) \quad (9)$$

Rather than only determining the best-matching neuron bm and modifying its weights, the neuron sbm with the second highest activation that fulfils (4) is

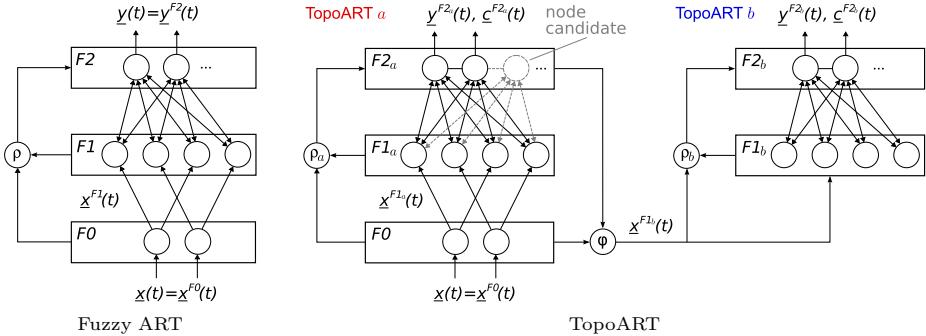


Fig. 1. Comparison of Fuzzy ART and TopoART. TopoART consists of two Fuzzy ART-like components using the same input layer. Furthermore, the F_2 nodes of TopoART are connected by edges defining a topological structure. In order to reduce its sensitivity to noise, TopoART evaluates the benefit of neurons (node candidates) before they are fully incorporated.

adapted as well (10). Here, β_{sbm} should be chosen smaller than 1, as neuron sbm – in contrast to neuron bm – is only intended to partly learn $\underline{x}^{F1}(t)$.

$$\underline{w}_{sbm}^{F2}(t+1) = \beta_{sbm}(\underline{x}^{F1}(t) \wedge \underline{w}_{sbm}^{F2}(t)) + (1 - \beta_{sbm})\underline{w}_{sbm}^{F2}(t) \quad (10)$$

As a result of this procedure, the insensitivity to noise is further increased, since the categories are more likely to grow in relevant areas of the input space.

If $\phi=1$ and $\beta_{sbm}=0$, inserted nodes immediately become permanent, all input vectors are propagated to TopoART b , and only the weights of the best-matching neuron bm are adapted during a learning cycle. In this case, the categories of TopoART a and TopoART b equal the categories of Fuzzy ART networks trained in fast-learning mode with the vigilance parameters ρ_a and ρ_b , respectively.

In order to enable TopoART to learn topologies, a lateral connection or rather edge between the neurons bm and sbm is created, if a second best-matching neuron can be found. These edges define a topological structure. They are not used for activating other neurons. If the neurons bm and sbm have already been connected by an edge, it remains unchanged, since the edges do not possess an age parameter in contrast to the edges in ESOINN, SOINN, and GNG. They are only removed, if one of the adjacent neurons is removed. As a consequence, edges between permanent nodes are stable. But new edges can still be created. This mechanism constitutes an extension of Fuzzy ART's solution to the stability-plasticity dilemma, which enables the representation of new input while retaining the already learnt representations.

In order to refine the representation of TopoART a by means of TopoART b , ρ_b should be higher than ρ_a . Within the scope of this paper, ρ_b is determined according to (11), which diminishes the maximum category size S^{\max} by 50%.

$$\rho_b = \frac{1}{2}(\rho_a + 1) \quad (11)$$

In this way, TopoART *b* learns a more detailed representation which is less influenced by noise. Connections between categories of TopoART *a* can be split in TopoART *b* resulting in a hierarchical representation of the input data.

The output $\underline{y}(t)$ of both TopoART components can be computed in a similar way as with Fuzzy ART. In addition, the connected components or rather clusters are labelled (cf. [12] and [13]): Starting from an unlabelled neuron, all connected neurons receive a specific label. Then, a new unlabelled neuron is searched for. This process is repeated until no unlabelled neurons remain. The vector $\underline{c}(t)$ provides the cluster labels of all F_2 neurons. For reasons of stability, only permanent nodes are considered for the computation of $\underline{y}(t)$ and $\underline{c}(t)$. By this, the clusters can grow and fuse, but they are prevented from shrinking.

4 Results

TopoART was evaluated using three different types of data: (i) stationary artificial data, (ii) non-stationary artificial data, and (iii) stationary real-world data. As stationary artificial input distribution, a dataset copying the one used for the evaluation of SOINN [12] was chosen (see Fig. 2(a)). It comprises two Gaussian components (A and B), two ring-shaped components (C and D), and a sinusoidal component (E) composed from three subcomponents (E1, E2, and E3). Each component encompasses 18,000 individual samples. Additionally, the input distribution includes uniformly distributed random noise amounting to 10% of the total sample number (10,000 samples).

This dataset was used to train a SOINN system. The results of both of its layers are depicted in Fig. 2(b) and Fig. 2(c), respectively. The values for λ , age_{dead} , and c were selected in such a way that results comparable to those published in [12] were achieved. In contrast, the settings for α_1 , α_2 , α_3 , β , and γ were directly adopted from [12] for both layers ($\frac{1}{6}$, $\frac{1}{4}$, $\frac{1}{4}$, $\frac{2}{3}$, $\frac{3}{4}$).

Figure 2 shows that SOINN was able to create a hierarchical representation of the input distribution: The two clusters in layer 1 were refined in layer 2 which

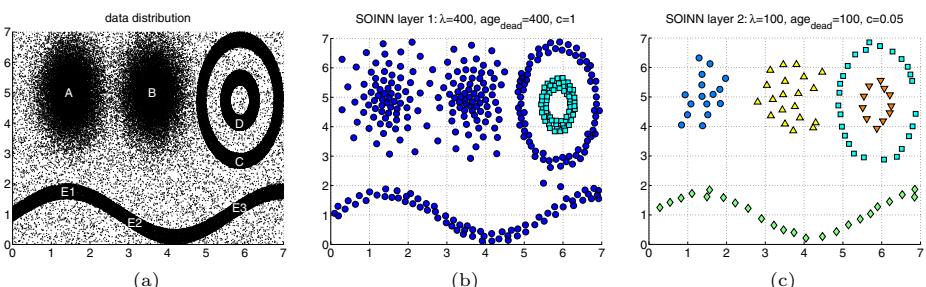


Fig. 2. Input distribution and clustering results of SOINN. The input distribution (a) is successfully clustered by SOINN. Here, the representation is refined from layer 1 (b) to layer 2 (c). Reference vectors belonging to the same cluster share a common colour and symbol.

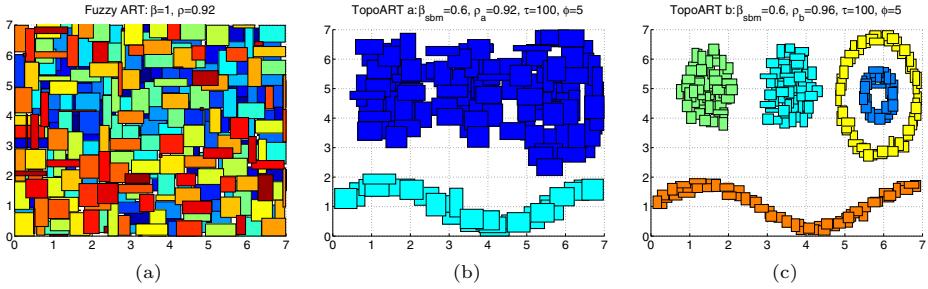


Fig. 3. Comparison of the ART networks’ results. Fuzzy ART (a) learnt rectangular categories covering virtually the complete input space. In contrast, TopoART learnt a noise insensitive representation in which the categories have been summarised to arbitrarily shaped clusters. The representation of TopoART *a* (b) was further refined by TopoART *b* (c). Here, all categories of an individual cluster are painted with the same colour.

distinguishes five clusters. Additionally, the second layer reduced the influence of noise.

The dataset shown in Fig. 2(a) cannot be used directly to train a Fuzzy ART or a TopoART network. Rather, all samples have to be scaled to the interval $[0, 1]$ so as to render them compatible with complement coding.

As Fuzzy ART constitutes the basis of TopoART, it was analysed first. For comparison reasons, β was set to 1 like β_{bm} in TopoART. ρ was selected in such a way as to roughly fit the thickness of the elliptic and sinusoidal components of the input distribution. As this network does not possess any means to decrease the sensitivity to noise, virtually the whole input space was covered by categories (see Fig. 3(a)).

In contrast to Fuzzy ART, both TopoART components created representations reflecting the relevant regions of the input distribution very well (see Figs. 3(b) and 3(c)). This is remarkable since the value of ρ_a was equal to the value of the vigilance parameter ρ of the Fuzzy ART network. Similar to the layers of SOINN, the representation of TopoART was refined from TopoART *a* to TopoART *b*: While TopoART *a* comprises two clusters, TopoART *b* distinguishes five clusters corresponding to the five components of the input distribution. By virtue of the filtering of samples by TopoART *a* and due to the fact that ρ_b is higher than ρ_a (cf. (11)), the categories of TopoART *b* reflect the input distribution in more detail. This property is particularly useful, if small areas of the input space have to be clustered with high accuracy. Here, TopoART *a* could filter input from other regions and TopoART *b* could create the desired detailed representation.

The learning of non-stationary data was also investigated using the input distribution shown in Fig. 2(a). But now, input from different regions was presented successively (see Fig. 4).

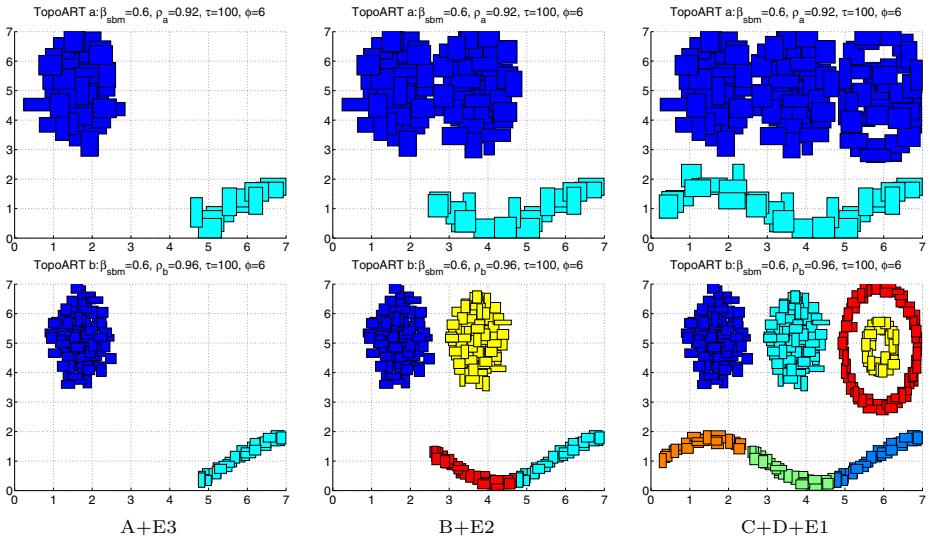


Fig. 4. Training of TopoART with non-stationary artificial data. A TopoART network was consecutively trained using three different input distributions: A+E3, B+E2, and C+D+E1. Each column shows the results after finishing training with data from the given regions. All categories of an individual cluster are painted with the same colour.

Figure 4 shows that both components of TopoART incrementally learnt the presented input. Already created representations remained stable when the input distribution changed. Similar to the stationary case, TopoART *b* performed a refinement of the representation of TopoART *a*. But here, the sub-regions E1, E2, and E3 were separated, since the corresponding input samples were presented independently and could not be linked. TopoART *a* was able to compensate for this effect, as its lower vigilance parameter ρ_a allowed for larger categories which could form connections between the sub-regions.

Finally, a dataset originally used to investigate methods for the direct imitation of human facial expressions by the user-interface robot iCat was applied [14]. From this dataset, the images of all 32 subjects (12 female, 20 male) that were associated with predefined facial expressions were selected, which resulted in a total of 1783 images. These images had been acquired using two lighting conditions: daylight and artificial light. They were cropped, scaled to a size of 64×64 pixels, and successively processed by principal component analysis keeping 90% of the total variance (cf. Fig. 5(a)).

After training a TopoART network with these data, the resulting clusters were compared to the partitionings based on labels reflecting the subjects and the lighting conditions. Here, two standard measures were used: the Jaccard coefficient J and the Rand index R [15]. Both provide values between 0 and 1, with higher values indicating a higher degree of similarity. As always more

than one criterion (subjects, lighting conditions, facial expressions, gender, usage of glasses, etc.) for partitioning these images influences the clustering process, values considerably lower than 1 may also indicate similarity. The splitting of clusters caused by these additional criteria, for example, results in a decrease of the Jaccard coefficient.

Due to the filtering process controlled by ϕ , several input samples were regarded as noise and excluded from cluster formation. Within the scope of the evaluation, they were associated with the cluster containing the permanent node with the highest activation. Since the original activation (3) depends on the category size, an alternative activation (12) was used to analyse the formed clusters.² It constitutes a normalised version of the city block distance between an input sample and the respective category, which was successfully applied in [3] and [16].

$$z_i^{F2}(t) = 1 - \frac{|(\underline{x}^{F1}(t) \wedge \underline{w}_i^{F2}(t)) - \underline{w}_i^{F2}(t)|_1}{d} \quad (12)$$

Figure 5 shows the results for both TopoART components. Here, the parameter τ was adopted from the previous experiments; β_{sbm} , ϕ , and ρ_a were iterated over representative values from the relevant intervals and selected in such a manner as to maximise the Jaccard coefficient for the partitioning according to the subjects. Then β_{sbm} and ϕ were fixed, while the influence of ρ_a is illustrated. Here, no results for $\rho_a < 0.75$ are displayed, since a further decrease of ρ_a does not cause any changes.

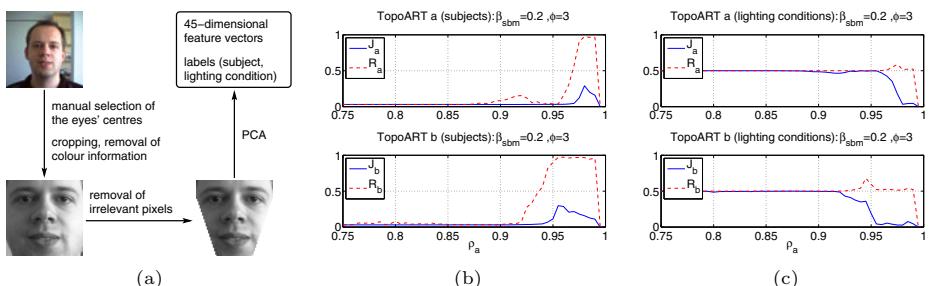


Fig. 5. Clustering of stationary real-world data. After preprocessing facial images (a), they were clustered by a TopoART network. Then, it was analysed how accurately the resulting clusters reflect the different subjects (b) and the two lighting conditions (c).

According to Fig. 5(b), the partitioning with respect to the 32 subjects is best represented if ρ_a is high. Here, J_b and R_b increase for lower values of ρ_a than J_a and R_a , which reflects the finer clustering of TopoART b. In contrast, the partitioning according to the two lighting conditions (cf. Fig. 5(c)) is impaired for high values of ρ_a , which is indicated by the decrease of J_a and J_b . The

² The original activation (3) was still utilised for training.

more detailed representation of TopoART b amplifies this effect. Choosing ρ_a appropriately, e.g. $\rho_a=0.96$, the representations of both partitionings can be combined by a single TopoART network. In this case, TopoART a represents the coarser partitioning with respect to the lighting conditions and TopoART b the finer partitioning according to the subjects.

For the results given in Fig. 5, the relevant parameters were selected in such a way as to maximise the Jaccard coefficient for the partitioning according to the subjects. Hence, the representation of the lighting conditions is not optimal. If the parameters are optimised to provide the maximum Jaccard coefficient for this partitioning, the following maximum values are reached: $J_a=0.779$ and $J_b=0.671$. This confirms the previous results according to which TopoART a creates a less detailed representation of the input data which is more suited to represent the two lighting conditions.

5 Conclusion

TopoART – the neural network proposed in this paper – successfully combines properties from ART and topology learning approaches: The categories originating from ART systems are connected by means of edges. In this way, clusters of arbitrary shapes can be formed. In addition, a filtering mechanism decreases the sensitivity to noise. Similar to SOINN, representations exhibiting different levels of detail are formed. But TopoART enables parallel learning at both levels requiring only 4 parameters ($\beta_{sbm}, \phi, \rho_a, \tau$) to be set, which constitutes a reduction of 75% compared to SOINN. Moreover, representations created by TopoART are stable.

Although the number of parameters to be set is small, they must be chosen using some kind of prior knowledge about the input data distribution. This task is very difficult, in particular, as TopoART is trained on-line. In order to solve this problem, the hierarchical structure of TopoART can be exploited, since it provides alternative clusterings of the input data distribution. By means of interaction during the learning process, these clusterings could be evaluated with respect to the current task or other criteria.

The capability of TopoART to capture hierarchical relations and the topology of presented data might be of interest for numerous tasks, e.g. the representation of complex sensory and semantic information in robots. In principle, TopoART could even be extended to a multi-level structure that captures hierarchical relations more comprehensively.

Acknowledgements. This work has partly been supported by the German Research Foundation (DFG) within the Center of Excellence ‘Cognitive Interaction Technology’, Research Area D (Memory and Learning). The author particularly wishes to thank Sina Kühnel and Marc Kammer for many fruitful discussions during the preparation of this manuscript.

References

1. Vigdor, B., Lerner, B.: Accurate and fast off and online fuzzy ARTMAP-based image classification with application to genetic abnormality diagnosis. *IEEE Transactions on Neural Networks* 17(5), 1288–1300 (2006)
2. Goerick, C., Schmüdderich, J., Bolder, B., Janßen, H., Gienger, M., Bendig, A., Heckmann, M., Rodemann, T., Brandl, H., Domont, X., Mikhailova, I.: Interactive online multimodal association for internal concept building in humanoids. In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pp. 411–418 (2009)
3. Tscherepanow, M., Jensen, N., Kummert, F.: An incremental approach to automated protein localisation. *BMC Bioinformatics* 9(445) (2008)
4. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297 (1967)
5. Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantizer design. *IEEE Transactions on Communications COM-28*, 84–95 (1980)
6. Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics* 43(1), 59–69 (1982)
7. Fritzke, B.: A growing neural gas network learns topologies. In: *Neural Information Processing Systems*, pp. 625–632 (1994)
8. Grossberg, S.: Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science* 11, 23–63 (1987)
9. Anagnostopoulos, G.C., Georgopoulos, M.: Hypersphere ART and ARTMAP for unsupervised and supervised incremental learning. In: *Proceedings of the International Joint Conference on Neural Networks*, vol. 6, pp. 59–64 (2000)
10. Anagnostopoulos, G.C., Georgopoulos, M.: Ellipsoid ART and ARTMAP for incremental clustering and classification. In: *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 1221–1226 (2001)
11. Carpenter, G.A., Grossberg, S., Rosen, D.B.: Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks* 4, 759–771 (1991)
12. Furao, S., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. *Neural Networks* 19, 90–106 (2006)
13. Furao, S., Ogura, T., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks* 20, 893–903 (2007)
14. Tscherepanow, M., Hillebrand, M., Hegel, F., Wrede, B., Kummert, F.: Direct imitation of human facial expressions by a user-interface robot. In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pp. 154–160 (2009)
15. Xu, R., Wunsch II, D.C.: Clustering. Wiley–IEEE Press (2009)
16. Tscherepanow, M.: Image analysis methods for location proteomics. PhD thesis, Faculty of Technology, Bielefeld University (2008)

Policy Gradients for Cryptanalysis

Frank Sehnke¹, Christian Osendorfer¹, Jan Sölter²,
Jürgen Schmidhuber^{3,4}, and Ulrich Rührmair¹

¹ Faculty of Computer Science, Technische Universität München, Germany

² Faculty of Biology, Freie Universität Berlin, Germany

³ Istituto Dalle Molle di Studi sull’Intelligenza Artificiale, Lugano, Switzerland

⁴ Faculty of Computer Science, Università della Svizzera italiana,
Lugano, Switzerland

Abstract. So-called Physical Unclonable Functions are an emerging, new cryptographic and security primitive. They can potentially replace secret binary keys in vulnerable hardware systems and have other security advantages. In this paper, we deal with the cryptanalysis of this new primitive by use of machine learning methods. In particular, we investigate to what extent the security of circuit-based PUFs can be challenged by a new machine learning technique named Policy Gradients with Parameter-based Exploration. Our findings show that this technique has several important advantages in cryptanalysis of Physical Unclonable Functions compared to other machine learning fields and to other policy gradient methods.

1 Introduction

Background on Physical Unclonable Functions. Physical Unclonable Functions (PUFs) are emerging recently as a powerful alternative to standard, mathematically based cryptography and security [1], [2]. In a nutshell, a PUF is a physical system S with a unique, partly disordered fine structure that depends on uncontrollable manufacturing variations. The system can be exposed to external stimuli or “challenges”. It reacts by returning so-called “responses”, whose value depends on said manufacturing variations.

Two typical applications of (Strong) PUFs are identification and key exchange scenarios. In these settings, PUFs have two advantages: (i) They avoid the storage of secret binary keys in vulnerable hardware systems, from which the keys can potentially be extracted by invasive attacks or viruses. (ii) They avoid the usual, unproven number theoretic assumptions that plague mathematical cryptography (albeit they rest on other assumptions). In other words, secure PUFs can create a new, advantageous form of cryptography in several application scenarios [1], [3], [4].

Machine Learning Attacks on PUFs. It has been realized relatively early in the history of PUFs [5] that Machine Learning (ML) techniques are a natural and also a very powerful tool to challenge the security of strong PUFs. In typical PUF applications, an adversary will be able to obtain a significant number

of challenges and corresponding responses (so-called challenge-response-pairs or CRPs) of the PUF. He can obtain the CRPs either by eavesdropping on communication protocols, or by gaining physical access to the PUF for a limited time period and measuring many CRPs himself. He can then feed these CRPs into an ML algorithm, interpreting the challenges as input and the responses as output of an unknown, to-be-learned function. If successfully trained, the algorithm will later predict the PUF’s responses with high probability, thereby breaking the security of the PUF, and of all protocols derived from it.

Our Contributions. This paper investigates to which extent the currently published electrical PUFs are susceptible to recent Policy Gradient (PG) methods. We show that small or medium size instances of almost all candidates of electrical Strong PUFs can be attacked well by a recent PG method called Policy Gradients with Parameter-based Exploration (PGPE) [6]. Our investigations show that PGPE is a particularly general and an efficient ML method for the cryptanalysis of electrical Strong PUFs. First of all, they merely require a parametric model of the attacked PUF, which is usually easy to identify. Contrary to that, the ML methods from the Reinforcement Learning (RL) or Supervised Learning domain that were applied to PUF cryptanalysis so far [7] all required differentiable models. Such models are much harder to find, and sometimes may not even exist.

Secondly, PGPE is faster and more reliable in attacking PUFs than population based heuristics (Evolution Strategies (ES) [8]), which were applied for cryptanalysis in a recent other publication of our group [7]. ES is the only other known ML strategy capable of attacking PUFs via a merely parametric model.

Organization of the Paper. The paper is organized as follows. In section 2 we define the investigated Arbiter PUF architectures. In section 3 we describe the population based heuristic used, namely Evolution Strategies (ES) and PGPE. Section 4 gives the results we obtained, in particular the obtained prediction rates plus the required CRPs and computation times for each ML method on each examined PUF. Section 5 summarizes the paper and discusses conclusions of our work.

2 Physical Unclonable Functions and Arbiter PUFs

As mentioned briefly in the introduction, a PUF is a physical system S with a unique, partly disordered fine structure that depends on uncontrollable manufacturing variations. The special security features of a (Strong) PUF S are the following: (i) Due to the partly random fine-structure of S , which should be beyond the control of its manufacturer, it must be impossible to fabricate a second physical system S' which has the same challenge-response behavior as S . (ii) Due to the complicated internal interactions of S , it must be impossible to devise a computer program that correctly predicts the response to a given challenge with high probability. This should hold even if many challenge-response pairs (CRPs) of S were known.

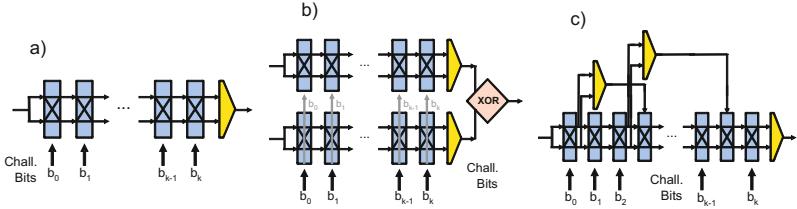


Fig. 1. Illustration of the architectures of a Standard Arbiter PUF (a), XOR Arbiter PUF (b) and Feed Forward Arbiter PUF (c). The challenge bits b_i at each stage decide if the two incoming signals propagate in parallel through the stage, or if their paths are crossed. All signal paths have slightly different run time properties due to uncontrollable, small fabrication variations. An arbiter element depicted as yellow at the end of the Standard Arbiter PUF (a) decides which of the two signals arrived first, and correspondingly outputs 0 or 1. In an XOR Arbiter PUF (b), the output of several Standard Arbiter PUFs is XORed. In FF Arbiter PUFs, signals at earlier stages of the circuit are fed into an arbiter element, whose output is applied as external bit at later stages of the circuit.

Together, the two conditions imply that the responses of S can be evaluated correctly only by someone who has got direct physical access to the single, unique system S . The validity of this assertion is essential for the security of all PUF-based protocols and schemes. But exactly this assertion can be challenged by ML techniques: A successfully trained ML algorithm can imitate a PUF's responses numerically, and can be copied and distributed at will.

All electrical PUFs analyzed in this paper have some common characteristics (see figure ①): The state of some switches is configured by a challenge vector C (with the i^{th} component encoding the state of the i^{th} switch), leading to pairs of unique propagation paths for an electric signal. The resulting propagation delay difference Δ between the pairs of paths is then further transformed by an arbiter gate (respectively combined arbiter and Xor gates) to a binary response t . Assuming that the overall propagation delay of a path is just the sum of the constant propagation delays of its constituent sub paths, Gassend et al. established a parametric linear model for the propagation delay difference ⑨. In a compact notation the model is given by

$$\Delta = \mathbf{w}^T \boldsymbol{\Phi} \quad (1)$$

where \mathbf{w} and $\boldsymbol{\Phi}$ are of dimension $k + 1$. The parameter vector \mathbf{w} encodes the delays for the subcomponents in the stages, whereas the feature vector $\boldsymbol{\Phi}$ is solely a function of the applied k -bit challenge C ⑤ ⑩ ⑪.

As shown in ⑤, the set of all possible linear propagation difference delay models (eq. ⑩) covers the characteristic of real PUF instances sufficiently well, such that each PUF instance can be assigned a model instance with its response prediction error in the range of the PUFs real-world stability. Therefore in this paper algorithms are presented which determine the suitable parameters \mathbf{w} provided that an adequate solution is contained in the set of linear propagation

delay models. That is, the algorithms are evaluated by applying them to data generated by the linear model itself with the sub delays drawn from a Gaussian distribution [11]. The detailed models for each PUF variant are explained in the subsections of section 4.

3 Employed Machine Learning Methods

Evolution Strategies. We chose Evolution Strategies as a benchmark for PGPE because they have been used in earlier publications by our group [7]. Up to now, they were the only ML method that was, at least in principle, applicable to all known electrical PUFs, since they merely required a parametric model of the PUF.

To attack PUFs with ES, an individual in the ES-population is given by a concrete instantiation of the runtime delays in a PUF (or by the vector \mathbf{w} from equation (1)). The environmental fitness is determined by how well this individual (re-)produces the correct CRPs of the target PUF as output. The outputs of the individual are computed by a linear additive delay model from its subdelays (or from \mathbf{w}), and are compared to several known outputs of the target PUF structure. We used a standard implementation of ES with the ES standard meta-parameters [12]: Population size of (6,36), comma-best-selection, and a global mutation operator with $\tau = \frac{1}{\sqrt{(n)}}$.

Policy Gradients with Parameter-based Exploration. In what follows, we briefly summarize [6] and [13], outlining the derivation that leads to PGPE. We give a short summary of the algorithm as far as it is needed for the rest of the paper. We assume that every executed episode or role out produces a scalar reward r . In this setting, the goal of reinforcement learning is to find the parameters θ that maximize the agent's expected reward

$$J(\theta) = \int_H p(h|\theta)r(h)dh \quad (2)$$

An obvious way to maximize $J(\theta)$ is to find $\nabla_\theta J$ and use it to carry out gradient ascent. Noting that the reward for a particular history is independent of θ , and using the standard identity $\nabla_x y(x) = y(x)\nabla_x \log y(x)$, we can write

$$\nabla_\theta J(\theta) = \int_H \nabla_\theta p(h|\theta)r(h)dh = \int_H p(h|\theta)\nabla_\theta \log p(h|\theta)r(h)dh \quad (3)$$

PGPE explores the search space by a probability distribution over the parameters θ , where ρ are the parameters determining the distribution over θ . The parameter gradient is therefore estimated by direct parameter perturbations, without having to backpropagate any derivatives, which allows the use of non-differentiable controllers or models (unlike standard PG methods). The expected reward with a given ρ is

$$J(\rho) = \int_\Theta \int_H p(h, \theta|\rho)r(h)dhd\theta. \quad (4)$$

Under the notion of several conditional independencies (given the details from [6]) and by referring to sampling methods, we get:

$$\nabla_{\rho} J(\rho) \approx \frac{1}{N} \sum_{n=1}^N \nabla_{\rho} \log p(\theta|\rho) r(h^n) \quad (5)$$

Sampling is done by first choosing θ from $p(\theta|\rho)$, then running the agent to generate h from $p(h|\theta)$. We assume that ρ 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 θ . In this case this assumption is especially useful because the fabrication variances of the Arbiter PUFs are around a known μ with an also known σ and are very well normal distributed and the delays in the PUF architecture are independent.

Some rearrangement gives the following forms for the derivative of $\log p(\theta|\rho)$ with respect to μ_i and σ_i :

$$\nabla_{\mu_i} \log p(\theta|\rho) = \frac{(\theta_i - \mu_i)}{\sigma_i^2} \quad \nabla_{\sigma_i} \log p(\theta|\rho) = \frac{(\theta_i - \mu_i)^2 - \sigma_i^2}{\sigma_i^3}, \quad (6)$$

which can then be substituted into (5) to approximate the μ and σ gradients.

We used the standard implementation of PGPE with the PGPE standard meta-parameters [6]: 2-Sample Symmetric Sampling, starting standard deviation for exploration as the standard deviation assumed for the PUFs and step sizes of 0.2 and 0.1 for the parameter and the sigma update. We also applied the usual reward normalization for PGPE.

4 Results

We will now discuss the results that we achieved in the application of the above machine learning techniques to the currently known electrical PUFs. If not stated differently, as the training data underlying the experiments we used a set of 50,000 CRPs with random subsets of 2,000 CRPs for the evaluation step of the individuals. These CRPs were generated on the basis of a linear additive delay model. The subdelays in the stages were drawn standard normal distributed.

4.1 Standard Arbiter PUF

Model. If we take the linear delay model from section 2 into account with respect to eq. II, the output t of this basic type of suggested electrical PUFs, the Standard Arbiter PUF (Arb-PUF), is determined by the sign of the propagation delay difference Δ .

Results. In all cases we have been able to learn the PUFs with prediction rates above 99% in 20,000 evaluations. Table II shows that the need of evaluations seems to grow only linearly with the number of bits. When comparing PGPE and ES, we observe that PGPE after the same number of evaluations achieves a remaining prediction error that is smaller by a factor of 5. Further, PGPE performs computationally about 4 times faster on this type of Arbiter PUF.

Table 1. The evaluations needed to achieve an average prediction rate of 90% and 95% with ES and PGPE. “E” marks the columns with the average evaluations, while “E/Bit” marks the columns that shows the evaluations needed per number of bits.

ES on Arb-PUFs				PGPE on Arb-PUFs					
Bit	90%		95%		Bit	90%		95%	
	E	E/Bit	E	E/Bit		E	E/Bit	E	E/Bit
16	446	27.88	720	45.00	16	118	7.38	190	11.88
32	878	27.44	1530	47.81	32	219	6.84	384	12.00
64	1879	29.36	3589	56.08	64	467	7.30	834	13.03
128	4230	33.05	9480	74.06	128	1080	8.44	1890	14.77

4.2 XOR Arbiter PUF

In this experiments we used random subsets of 8,000 CRPs for the evaluation step of the individuals for all XOR Arbiter PUF (XOR-PUF) experiments.

Model. One possibility to strengthen the resilience of arbiter architectures against machine learning is to employ l individual Arb-PUFs, each with k stages. The same k -bit challenge $C = b_1 \dots b_k$ is applied to each of these Arb-PUF, and their individual outputs t_i are XORed (i.e. added modulo 2) in order to produce a global output t_{XOR} [14] (see Fig. II b). We denote such an architecture as l -XOR-PUF. This builds a “needle in a haystack” search space that is very challenging for ES and PGPE.

Results. Figures 2 and 3 show the best of a total of 10 runs on each XOR-PUF that have been conducted. Table 2 shows the needed evaluations to achieve a prediction rate of 90% and the fraction of runs that have achieved this prediction rate in the given maximal number of evaluation steps. Clearly the number of evaluations needed grows more than linearly for ES and PGPE. We speculate on the basis of the obtained data that the growth rate is higher degree polynomial, but stress that due to noise and the small database, a definite and final conclusion

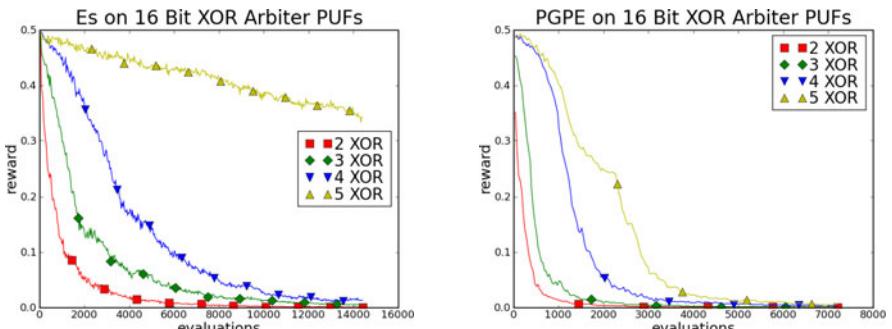


Fig. 2. The best of 10 runs on each XOR-PUF architecture with a 16 bit input vector

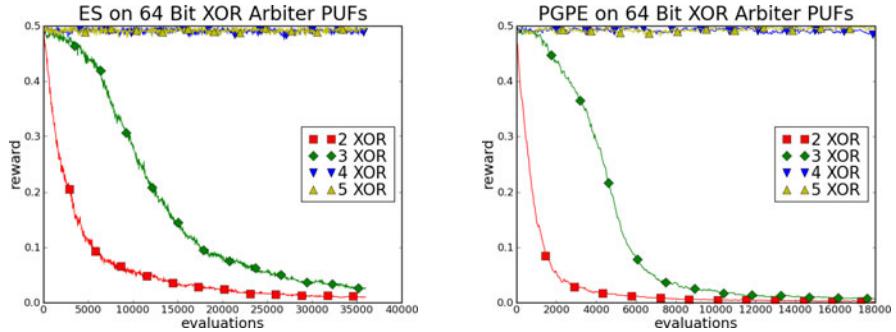


Fig. 3. The best of 10 runs on each XOR-PUF architecture with a 64 bit input vector

is difficult. The fraction of runs that succeed in predicting the PUF with a suitable rate drops strongly with the number of XOR-inputs. However, as shown, in all cases up to 3 XOR-inputs we have been able to successfully learn the PUFs with rates better than 10% with both ML methods. Our experiments show that the XOR-PUF can be broken up to 3 XORS and challenge length of 64 bit. When comparing PGPE and ES, we observe that PGPE performs computationally about 3 times faster on this type of PUF. Also the success rate drops less for PGPE, so PGPE seems slightly more reliable on breaking this kind of PUF as can be seen nicely in Figure 2.

4.3 Feed Forward Arbiter PUF

The Feed Forward Arbiter PUF (FF-PUF) is the most important type of PUF for this paper. Their models are, in general, not differentiable, and are therefore not prone to supervised learning and to standard PG methods. In [7], we showed that FF-PUFs are prone to attacks based on Evolutionary Algorithms. In this section, we show that PGPE is a better alternative to ES in breaking this PUF and therefore opens up the RL domain for attacking this type of PUF. The exact architecture of the chosen FF-PUF structures (length of loops, start and

Table 2. The number of evaluations needed to achieve a prediction rate of 90% and the rate of runs that achieved this prediction rate in the given maximal number of evaluations with ES and PGPE. "E" marks the columns with the evaluations needed, while "Rate" marks the column that shows the rate of successful runs.

ES on XOR-PUFs					PGPE on XOR-PUFs				
XOR	E		Rate		XOR	E		Rate	
	16 Bit	64 Bit	16 Bit	64 Bit		16 Bit	64 Bit	16 Bit	64 Bit
2	1080	5508	100%	100%	2	315	1350	100%	100%
3	3031	17460	90%	50%	3	556	5620	50%	90%
4	5796	-	30%	0%	4	1690	-	40%	0%
5	-	-	0%	0%	5	2810	-	40%	0%

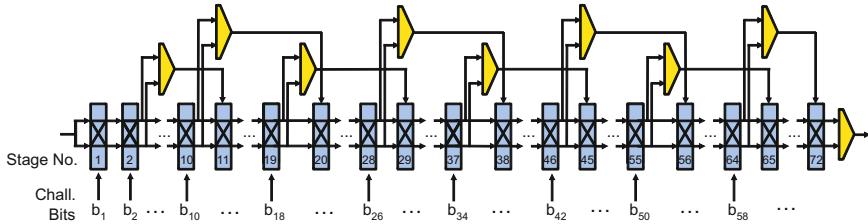


Fig. 4. The architecture of the FF-PUFs that we employed in our ML experiments, shown for the 8 FF-loop case

end point of loops) are shown in Figure 4. This architecture of symmetric and equally distributed loops seems the most natural way of placing the loops, and has also been used in earlier publications [15, 7]; investigations are on the way to consolidate that this is indeed the optimal, i.e. hardest to learn, architecture.

Model. FF-PUFs were introduced in [9, 15, 5] and further discussed in [11]. Their basic layout is similar to the architecture of Arb-PUFs. However, some of their multiplexers are not switched in dependence of an external challenge bit, but as a function of the delay differences accumulated in earlier parts of the circuit. Additional arbiter components evaluate these delay differences, and their output bit is fed into said multiplexers in a “feed-forward loop” (FF-loop). The number of loops as well as the starting and end point of the FF-loops are variable design parameters. Please note that a FF Arb-PUF with k -bit challenges $C = b_1 \cdots b_k$ and l loops has $s = k + l$ multiplexers or stages. The described dependency makes natural architecture models of FF Arb-PUFs not differentiable any more. This architecture generates an interesting multi-modal search space that is also challenging for ES and PGPE.

Results. Figure 5 shows the best of a total of 40 runs for ES and 10 runs for PGPE on each FF-PUF that have been conducted while varying the number of

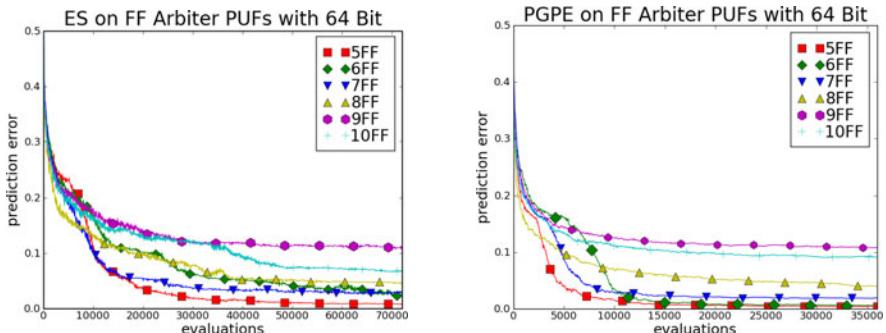


Fig. 5. The best of 40 runs (80 for 9+10 FF) with ES and the best of 10 runs (20 for 9+10 FF) with PGPE on each FF-PUF architecture

Table 3. The evaluations needed to achieve an prediction rate of 90% and 95% for the best run out of 40 (80 for 9+10 FF) for ES and out of 10 (20 for 9+10 FF) for PGPE. E stands for the needed number of evaluations, and E/FF symbolizes the needed evaluations divided by the number of FF loops. "Best" marks the columns with the best results after 72,000 evaluations (ES) and 36,000 evaluations (PGFPE).

FF	ES on FF-PUFs				PGPE on FF-PUFs					
	90%		95%		Best Result	90%		95%		Best Result
		E/FF		E/FF		E	E/FF	E	E/FF	
5	10200	2040	17100	3420	99.3%	2900	580	3730	746	99.6%
6	21200	3533	42300	7050	97.7%	7840	1307	9340	1557	99.5%
7	10100	1443	22700	3243	97.4%	4710	673	6580	940	98.2%
8	17600	2200	53100	6638	95.5%	4840	605	21350	2669	96.1%
9	-	-	-	-	89.2%	-	-	-	-	89.3%
10	37500	3750	-	-	93.4%	15480	1548	-	-	90.9%

FF-loops. Table 3 shows the evaluations needed to achieve an average prediction rate of 90% and 95%. As shown, in all cases we have been able to successfully learn the PUFs with rates close to 90%. Please note that our accuracy is significantly better than the stability of an in-silicon FF-Arbiter with 7 FF-loops while undergoing a temperature change of $45^{\circ}C$, which is only 90.16% [16]. The experiments show that the FF-Arbiter are definitely susceptible to attacks by ES and PGPE. When comparing PGPE and ES, we observe that PGPE performs computationally about 3 times faster on this type of Arbiter PUF.

5 Summary and Conclusion

We investigated the performance of the recently published Policy Gradient method PGPE in the cryptanalysis of electrical PUFs. We found that up to a medium level of size and complexity, essentially all currently known electrical PUFs are susceptible to attacks by this method. One particular advantage of PGPE in the cryptanalysis of circuit-based PUFs is, that it merely requires a parametric internal model of the PUF. In opposition to that, other ML methods that have been applied for PUF attacks require linearly separable or differentiable models, which can be hard to find, or may not even exist at all. Our results further reveal that PGPE significantly outperforms Evolution Strategies in the cryptanalysis of PUFs. ES was up to now the only other known ML method applicable to all existing electrical PUFs, since it also worked on the mere basis of a parametrizable model. Due to their broad applicability, and since they do not require hard-to-optimize differentiable or separable models, PGPE-based tests have the potential of becoming a standard security benchmark for circuit-based PUFs: ML curves obtained by PGPE on small instances can help us to judge and compare the security of various electrical PUF implementations.

References

1. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. *Science* 297(5589), 20–26 (2002)
2. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security-CCS, pp. 148–160 (2002)
3. Pappu, R.: Physical One-Way Functions. Phd thesis, MIT
4. Rührmair, U., Busch, H., Katzenbeisser, S.: Strong pufs: Models, constructions and security proofs, *Towards Hardware Intrinsic Security: Foundation and Practice*. Springer, Heidelberg (2010)
5. Lim, D.: Extracting Secret Keys from Integrated Circuits. Msc thesis, MIT (2004)
6. Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., Schmidhuber, J.: Parameter-exploring policy gradients. *Neural Networks* 23(4), 551–559 (2010)
7. Rührmair, U., Sehnke, F., Söltner, J., Dror, G., Stoyanova, V., Schmidhuber, J.: Machine learning attacks on physical unclonable functions. In: ACM Conference on Computer and Communications Security-CCS (2010) (to be published)
8. Schwefel, H.: *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., New York (1993)
9. Gassend, B., Lim, D., Clarke, D., Van Dijk, M., Devadas, S.: Identification and authentication of integrated circuits. *Concurrency and Computation: Practice & Experience* 16(11), 1077–1098 (2004)
10. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight secure PUFs. In: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, pp. 670–673. IEEE Press, Los Alamitos (2008)
11. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing techniques for hardware security. In: Proceedings of the International Test Conference (ITC), pp. 1–10 (2008)
12. Bäck, T.: *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA (1996)
13. Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., Schmidhuber, J.: Policy gradients with parameter-based exploration for control. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 387–396. Springer, Heidelberg (2008)
14. Suh, G., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th annual Design Automation Conference, vol. 14, ACM, New York (2007)
15. Lee, J., Lim, D., Gassend, B., Suh, G., Van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Proceedings of the IEEE VLSI Circuits Symposium, p. 176 (2004)
16. Lim, D., Lee, J., Gassend, B., Suh, G., Van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration Systems* 13(10), 1200 (2005)

Linear Projection Method Based on Information Theoretic Learning

Pablo A. Vera¹, Pablo A. Estévez¹, and Jose C. Principe²

¹ Department of Electrical Engineering and Advanced Mining Technology Center,
Universidad de Chile,
Casilla 412-3, Santiago, Chile

² CNEL, University of Florida

{pavera,pestevaz}@ing.uchile.cl, principe@cnel.ufl.edu

Abstract. A new unsupervised method for linear projection of multi-dimensional data based on Linsker's principle of maximum information preservation is proposed. The Quadratic Mutual Information (QMI) between the input X and the output Y is estimated, assuming a linear mapping. This estimation is made using a non-parametric quadratic divergence measure, without any assumption of data distribution. The results show that the 2D projections obtained with the proposed method are better than PCA projections in terms of cluster separability.

Keywords: Data Visualization, Mutual information, Renyi entropy, Linear Projection, PCA.

1 Introduction

Dimensionality reduction is a key topic in Pattern Recognition and Exploratory Data Analysis. The objective of dimensionality reduction is to project data belonging to a high dimensional space onto a space of lower dimensionality. This projection could be used as a pre-processing stage for a classifier, to compress data for storage or transmission, and for clustering or visualization purposes. In the latter case the projections are usually done onto a two-dimensional or 3D output space. This process can be done by feature transformation, where the original data is transformed in order to extract the most relevant components according to some criterion. The feature transformation can be linear or non-linear. Among the methods based on linear transformations are principal component analysis (PCA) [3], independent component analysis (ICA) and projection pursuit [19]. On the other hand, there are many nonlinear projection methods, among others Sammon's mapping [15], Isomap [17] and Oving [1].

The resulting projections could be assessed by quality mapping measures. Most existing quality measures are based on distance ranking and k-ary neighborhoods [8], which assess the preservation of geometrical properties of the data set. An alternative goal is to obtain visual projections that show points belonging to different clusters or classes well separated. VizRank [7] assesses data projections by estimating the quality of class separation by computing a k-nearest neighbor classifier on the projected data.

Information Theoretic Learning (ITL) [11][12] is a framework that allows extracting information directly from data samples. If a data set conveys information about a real-world event, the goal of ITL is to capture that information in the parameters of a learning machine. Usually the training criteria is based on entropy or mutual information (MI) maximization (minimization). Torkkola [18] proposed to find a linear transformation of features by maximizing the MI between class labels and the transformed features. An efficient non-parametric estimate of the Quadratic Mutual Information (QMI) is computed. MI takes into account higher-order statistics, not just second order as PCA or linear discriminant analysis. However, Torkkola's method is supervised with the aim of enhancing class separability.

In this paper, a new linear projection method based on QMI maximization is proposed. This method is non-parametric and unsupervised. It does not use any information about the class of the sample nor make any assumption about the data distribution. It aims at preserving the data distribution and enhancing the cluster separability.

2 Information Theoretic Learning

2.1 ITL Optimization Principles

Consider a parametric mapping $g : \Re^D \rightarrow \Re^d$ of a random variable X with N examples $\{x_i\} \in \Re^D$, with $d < D$, described by $y_i = g(x_i, W)$. Let Y be another random variable with N samples $y_i \in \Re^d$, and W a set of parameters. The goal of ITL is to choose the parameters W based on an information theoretic optimization principle, where the optimization is based only on samples x_i and y_i , and no assumption about their probability density functions (PDFs) is made. There are three well-known information optimization principles. Jayne's MaxEnt principle [4] finds the distribution that maximizes entropy. Kullback's MinXEnt [4] finds a distribution that minimizes a divergence measure in the probability space, such as the Kullback-Leibler (KL) divergence. Linsker's principle of maximum information preservation (Infomax) [9], maximizes the average mutual information between the input variable X and the output Y in the presence of noise. Since our goal is to map high-dimensional vectors onto a low-dimensional space, Linsker's Infomax principle will be used here.

2.2 Quadratic Mutual Information

The classic definition of MI is associated with Shannon's entropy [14]. Let X be a random variable which takes values x_1, x_2, \dots, x_N with probabilities p_k , $k = 1, 2, \dots, N$. Shannon's entropy measures the average amount of information conveyed by the event X . Principe et al. [12] proposed to combine Renyi's entropy [13] with Parzen windows [10], which is a PDF estimator based on a sum of kernels, each one placed on each sample. The differential Renyi's entropy of order 2 is defined as follows,

$$H_{R2}(X) = -\log \int f(x)^2 dx. \quad (1)$$

The Gaussian kernel is defined in a D -dimensional space as

$$G(x, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}x^T \Sigma^{-1} x\right), \quad (2)$$

where Σ is the covariance matrix, and $x \in \Re^D$. Let x_i and $x_j \in \Re^D$ be two samples, Σ_1 and Σ_2 covariance matrices for two Gaussian kernels. The Parzen window estimation for the PDF of a set of samples $x_i \in \Re^D, i = 1, \dots, N$ is

$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^N G(x - x_i, \sigma^2 I) \quad (3)$$

where the covariance matrix has been assumed diagonal $\sigma^2 I$. Using Renyi's quadratic entropy leads to a simple form to estimate the entropy directly from data samples,

$$H_{R2}(X) = -\log \left(\int \hat{f}(x)^2 dx \right) = -\log V(X), \quad (4)$$

$$V(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G(x_i - x_j, 2\sigma^2 I). \quad (5)$$

Given two random variables X and Y , with marginal PDFs $f(x)$ and $f(y)$, and joint PDF $f(x, y)$, the Quadratic Mutual Information (QMI) based on the Euclidean Difference of vectors inequality is defined as [12]:

$$QMI_{ED}(X, Y) = \int \int f(x, y)^2 dx + \int \int f(x)^2 f(y)^2 dxdy - 2 \int \int f(x, y) f(x) f(y) dxdy. \quad (6)$$

Replacing the corresponding PDFs for their Parzen window estimators, yields [12]:

$$QMI_{ED} = V_{ED} = V_J - 2V_C + V_M, \quad (7)$$

where:

$$\begin{aligned} V_J &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G(x_i - x_j, \sigma_x^2 I) G(y_i - y_j, \sigma_y^2 I) \\ V_C &= \frac{1}{N} \sum_{i=1}^N \left\{ \frac{1}{N} \sum_{j=1}^N G(x_i - x_j, \sigma_x^2 I) \cdot \frac{1}{N} \sum_{j=1}^N G(y_i - y_j, \sigma_y^2 I) \right\}. \\ V_M &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G(x_i - x_j, \sigma_x^2 I) \cdot \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G(y_i - y_j, \sigma_y^2 I) \end{aligned} \quad (8)$$

These three terms average pairwise interaction in different spaces. V_J is the average of the interaction in the joint space, V_C is the average of the cross-interaction between different spaces and V_M is the average in each marginal space.

2.3 Learning Algorithm

In this paper, for the sake of simplicity a linear mapping is assumed, $Y = W^T X$. This is a linear transformation from the input space to the output space, where each sample $x_i \in \Re^D$ has a projection $y_i = W^T x_i$ in the output space. W is a matrix of size $D \times d$ where each column defines a projection axis. A general rule to optimize the set of parameters W is to use a gradient ascent method as an iterative process to reach the optimum, starting from a random point. The parameters are updated iteratively as follows:

$$W(t+1) = W(t) + \eta \frac{\partial V_{ED}}{\partial (y_i - y_j)} \frac{\partial (y_i - y_j)}{\partial W}. \quad (9)$$

Notice that the derivative of V_{ED} is made with respect to $d_{ij} = y_i - y_j$. The reason why is because the QMI depends on the pairwise interaction, not just the particle position y_i .

Assuming a linear transformation, $Y = W^T X$, the derivative of d_{ij} with respect to W can be written as:

$$\frac{\partial (y_i - y_j)}{\partial W} = (x_i - x_j)^T. \quad (10)$$

Computing the derivative of QMI_{ED} is tantamount to compute the derivative of each term in (8) separately, yielding the following expressions:

$$\begin{aligned} \frac{\partial V_J}{\partial W} &= -\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N V_{ij}^x V_{ij}^y \frac{(y_i - y_j)}{\sigma_y^2} (x_i - x_j)^T \\ \frac{\partial V_C}{\partial W} &= -\frac{1}{N} \sum_{i=1}^N \left\{ \frac{1}{N} \sum_{j=1}^N V_{ij}^x \cdot \frac{1}{N} \sum_{j=1}^N V_{ij}^y \frac{(y_i - y_j)}{\sigma_y^2} (x_i - x_j)^T \right\} \\ \frac{\partial V_M}{\partial W} &= -\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N V_{ij}^x \cdot \sum_{i=1}^N \sum_{j=1}^N V_{ij}^y \frac{(y_i - y_j)}{\sigma_y^2} (x_i - x_j)^T. \end{aligned} \quad (11)$$

2.4 ITL-Based Linear Projection Method

Let us define X as a random variable with N samples $\{x_i \in \Re^D, i = 1 \dots N\}$. The goal is to find the projection $y_i \in \Re^d$ for each sample optimizing the parameters W of the linear projection maximizing $QMI_{ED}(X, Y)$. The transformation matrix W is constrained to be orthonormal in order to project to an orthogonal space. The simplest way of taking into account this constraint is to orthonormalize W after each update.

A pseudo-code of the proposed algorithm for data visualization, called Infomax-ITL (IITL), is as follows:

1. Initialize matrix W of size $D \times d$ randomly
2. Set the parameters σ_x , σ_y , using (13) and the learning rate η
3. Use a gradient ascent algorithm to optimize W by maximizing $V_{ED}(X, Y)$.
At each iteration:
 - Compute $\frac{\partial V_{ED}}{\partial W}$ using (11)
 - Update the weights using (9)
 - Constrain W to be orthonormal
 - Update the position of the projections using $y_i = W^T x_i$
 - Repeat until reaching stop criterion

The stop criterion can be set to a maximum number of iterations or a threshold of minimum variation on the value of $V_{ED}(X, Y)$ in a certain number of epochs.

2.5 Preprocessing for IITL

Before applying the IITL algorithm, a preprocessing step is performed by scaling data in [-1,1] and whitening the features. The whitening process transforms the data vectors x linearly to obtain a new data vector, \tilde{x} , whose components are uncorrelated and have unitary variance. Formally $E\{\tilde{x}\tilde{x}^T\} = I$ [2]. As kernel based methods usually use the same σ value for every dimension, the whitening process makes the estimation of the σ size more optimal and precise. A simple way to whiten data using the covariance matrix of the original data, Σ , is to do the following transformation [5]:

$$\tilde{x}_i = \Sigma^{-1/2} (x_i - \bar{x}). \quad (12)$$

3 Experiments

The proposed algorithm is compared with PCA using three different indicators of quality mapping and three datasets. The comparison with PCA is appropriate since both algorithms are global linear projectors, and the main difference is the functional to be optimized.

3.1 Datasets

Two datasets (Tetra and Hepta) were drawn from the Fundamental Clustering Problem Suite (FCPS)¹. In addition, the Pipeline dataset was taken from the Neural Computing Research Group at Aston University².

Tetra: It contains 400 samples from 4 spherical clusters in 3D.

Hepta: It contains 212 samples from 7 spherical clusters in 3D.

Pipeline: This synthetic data set models non-intrusive measurements on a pipeline transporting a mixture of oil, water and gas. There are 13 features for each of the 1000 samples, representing 3 classes.

¹ <http://www.uni-marburg.de/fb12/datenbionik/>

² <http://www.ncrg.aston.ac.uk/GTM>

3.2 Parameter Setting

The Silverman's rule-of-thumb [16], is used to estimate the kernel size, which for the multivariate d -dimensional case becomes

$$\sigma_S = \hat{\sigma} \left[\frac{4}{(2d+1)N} \right]^{\left(\frac{1}{d+4}\right)} \quad (13)$$

where $\hat{\sigma}^2 = d^{-1} \sum_i^d \sigma_{ii}$, and σ_{ii} are the diagonal elements of the sample covariance matrix. For the input space σ_x is estimated directly using the covariance matrix of the data. In our experiments, σ_y is computed using the covariance matrix of the input data but replacing the dimension of the input space by the dimension of the output space in rule (13). For a better convergence to a global maximum, the kernel size is annealed by decreasing the value of σ exponentially through the iterative process from an initial value σ^+ to a final value σ^- , as follows:

$$\sigma_i = \sigma_S \cdot \sigma^+ \left(\frac{\sigma^-}{\sigma^+} \right)^{\frac{i}{i_{max}}}. \quad (14)$$

Likewise the learning rate η is also annealed exponentially.

3.3 Quality Mapping Measures

A quality mapping index that measures the k-ary neighborhood preservation between the high- and low dimensional spaces is used, the so-called q_{nx} [8], which takes values in $[0, 1]$. The closer to one is this index the better is the neighborhood preservation. In order to assess data projections by estimating the quality of cluster separation the Dunn index [19] is computed. A large value of this index indicates the presence of more compact and well separated clusters. When the class labels are available, data projections can be assessed by using the accuracy of k-nearest neighbor (KNN) classifier on the projected data [7].

3.4 Simulation Results

Fig. 1 shows the data projections obtained with the proposed IITL method and PCA for the Tetra dataset. It can be observed that the four clusters are well separated in the IITL projection, while they overlap in the PCA projection. Fig. 2a contains the q_{nx} quality measure for both IITL and PCA projections using the Tetra dataset. A small index improvement is obtained when using IITL versus PCA, for less than 200 neighbors. For the tetra dataset the following parameters were used: {200 iterations, $\sigma^+ = 1.5$, $\sigma^- = 0.5$, $\eta^+ = 10^6$, $\eta^- = 10^5$ }. In Table II, the first two rows show the assessment of data projections for the Tetra dataset. Although the Dunn index value is very similar for both the IITL and PCA projections, the KNN classifier accuracy of the IITL projection surpasses the performance of the same classifier with PCA projections by 28%.

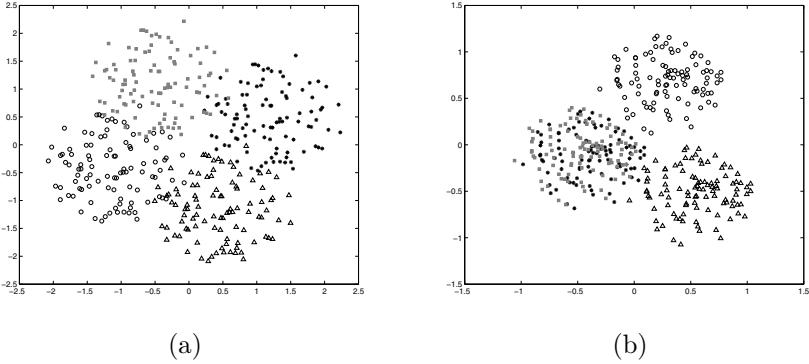


Fig. 1. Data visualization obtained for Tetra dataset using a) IITL and b) PCA

Fig. 3 shows the data projections obtained with IITL and PCA for the Hepta data set. A slightly better separation of clusters is obtained with the IITL projection. Fig. 2b contains the q_{nx} neighborhood preservation quality measure for both IITL and PCA projections of Hepta dataset. A small improvement is obtained in this index when using IITL in comparison with PCA, for less than 40 neighbors. For the Hepta dataset the parameters were set as $\{120$ iterations, $\sigma^+ = 1.5, \sigma^- = 0.5, \eta^+ = 10^5, \eta^- = 10^4\}$. From Table II, the Dunn index for the IITL projection is much higher than that of PCA projection, indicating a better cluster separation. The KNN classifier accuracy for both projections is 100%. Fig. 4 shows the data projections obtained with the proposed IITL method and PCA for the Pipeline dataset. It can be seen that the IITL projection obtains a better cluster separation, compared with the PCA projection. Fig. 2c contains the q_{nx} neighborhood preservation quality measure of both IITL and PCA projections for the Pipeline dataset. Contrary to the previous two examples, in this case PCA obtained a better neighborhood preservation when compared with IITL for any number of neighbors. For the Pipeline dataset the parameters were set as $\{150$ iterations, $\sigma^+ = 1.0, \sigma^- = 0.5, \eta^+ = 10^8, \eta^- = 10^7\}$. In Table II, the last two rows show that the Dunn index value and the KNN accuracy (99.7%) are higher for the IITL projection than for the PCA projection using Pipeline

Table 1. Assessment of data projections using Dunn index and KNN classifier

Dataset	Method	Dunn Index	k-NN Accuracy (%)
Tetra	IITL	0.0176	91.7
	PCA	0.0178	63.3
Hepta	IITL	0.3186	100
	PCA	0.0922	100
Pipeline	IITL	0.0352	99.7
	PCA	0.0035	86.7

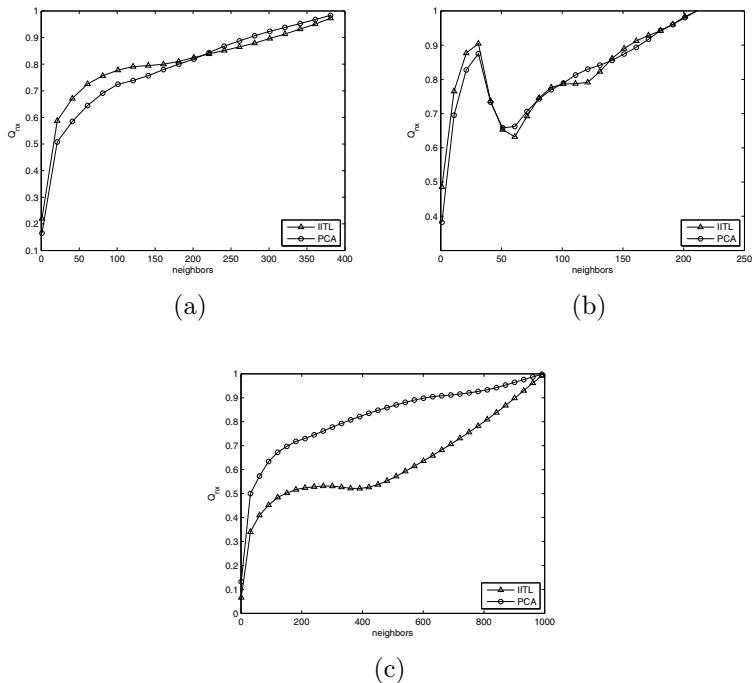


Fig. 2. Quality assessment of IITL and PCA projections using q_{nx} index for a) Tetra dataset, b) Hepta dataset, and c) Pipeline dataset

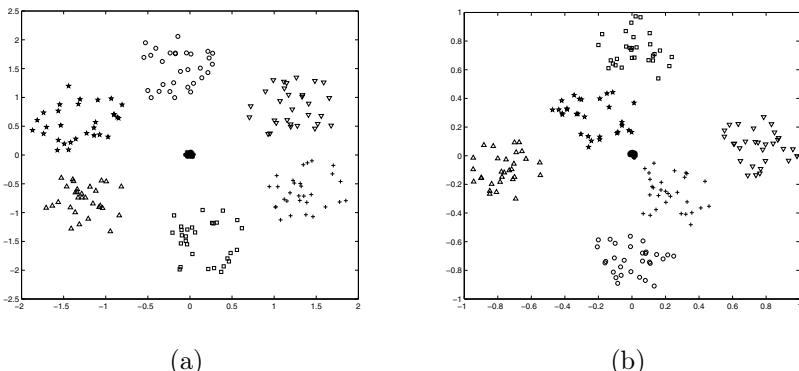


Fig. 3. Data visualization obtained for Hepta dataset using a) IITL and b) PCA

dataset. Fig. 5 shows the Dunn index and the KNN classification rate as a function of the kernel size for the Pipeline dataset. It can be observed that the best Dunn index and KNN classification rate is obtained when the kernel size is equal to Silvermann's kernel size.

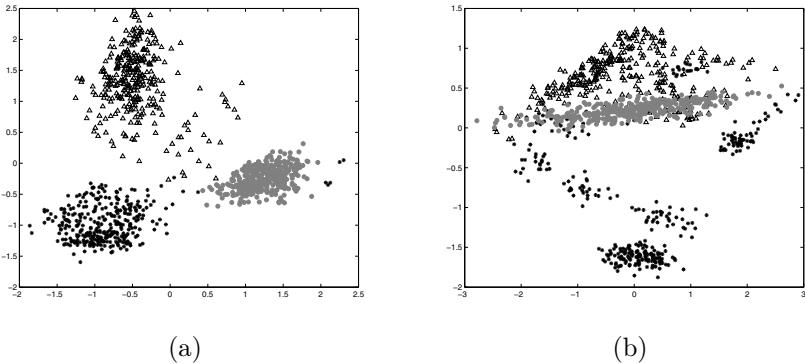


Fig. 4. Data projections for Pipeline dataset using a) IITL and b) PCA

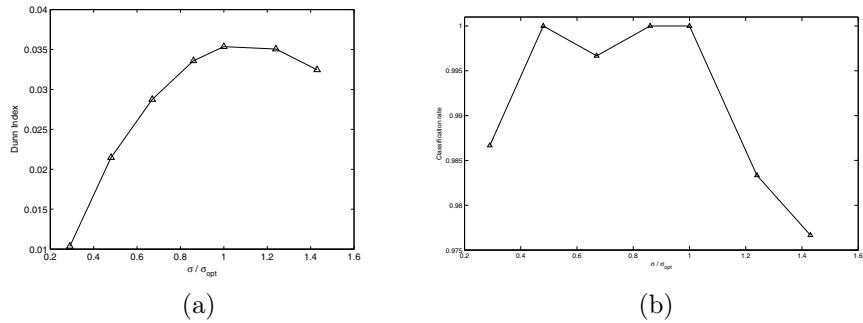


Fig. 5. a) Dunn index and b) KNN classification rate versus the kernel size for the Pipeline dataset

4 Conclusion

A new linear projection method based on the Infomax criterion has been proposed. The results show that IITL obtained better projections than PCA in terms of cluster (class) separability on three datasets. This result matches with the visual assessment of goodness of the data projections. The results on the Pipeline dataset show that it is possible to get higher cluster separability at the expense of lower neighborhood preservation. These promising results open the question of defining a more suitable index of goodness of data projections. In addition, this approach could be easily extended to the optimization of non-linear mappers.

Acknowledgment

This research was funded by Conicyt-Chile under grant Fondecyt 1080643.

References

1. Estevez, P.A., Figueroa, C.J.: Online Data Visualization Using the Neural Gas Network. *Neural Networks* 19, 923–934 (2006)
2. Hyvärinen, A., Oja, E.: Independent Component Analysis: Algorithms and Applications. *Neural Networks* 13, 411–430 (2000)
3. Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer, New York (2002)
4. Kapur, J., Kesavan, H.: Entropy Optimization Principles with Applications. Academic Press, New York (1992)
5. Kojadinovic, I.: Relevance Measures for Subset Variable Selection in Regression Problems Based on k-Additive Mutual Information. *Computational Statistics and Data Analysis* 49, 1205–1227 (2005)
6. Kullback, S.: Information Theory and Statistics. Dover Publications, New York (1968)
7. Leban, G., Zupan, B., Vidmar, G., Bratko, I.: VizRank: Data Visualization Guided by Machine Learning. *Data Mining and Knowledge Discovery* 13, 119–136 (2006)
8. Lee, J.A., Verleysen, M.: Quality Assessment of Nonlinear Dimensionality Reduction Based on K-ary Neighborhoods. *Journal of Machine Learning Research* 4, 21–35 (2008)
9. Linsker, R.: An Application of the Principle of Maximum Information Preservation to Linear Systems. In: Touretzky, D.S. (ed.) *Advances in Neural Information Processing Systems*, vol. 1, pp. 186–194. Morgan Kauffman, San Francisco (1989)
10. Parzen, E.: On the Estimation of a Probability Density Function and Mode. *Annals of Mathematical Statistics* 33, 1065–1076 (1962)
11. Principe, J.C.: Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives. Springer, Heidelberg (2010)
12. Principe, J.C., Fisher III, J.W., Xu, D.: Information Theoretic Learning. In: Haykin, S. (ed.) *Unsupervised Adaptive Filtering*, John Wiley, New York (2000)
13. Renyi, A.: On Meausures of Entropy and Information. In: 4th Berkeley Simp. Math. Stat. and Prob., vol. 1, pp. 547–561 (1961)
14. Shannon, C.E.: A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 379–423, 623–656 (1948)
15. Sammon, J.W.: A Nonlinear Mapping Algorithm for Data Structure Analysis. *IEEE Transactions on Computers* 18, 401–409 (1969)
16. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)
17. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 290, 2319–2323 (2000)
18. Torkkola, K.: Feature Extraction by Non-parametric Mutual Information Maximization. *Journal of Machine Learning Research* 3, 1415–1438 (2003)
19. Xu, R., Wunsch II, D.C.: Clustering. IEEE Press/John Wiley Inc. (2009)

Continuous Visual Codebooks with a Limited Branching Tree Growing Neural Gas

Marco Kortkamp¹ and Sven Wachsmuth^{1,2}

¹ Applied Informatics, Faculty of Technology

² Central Lab Facilities, CITEC

Universitätstraße 25, 33615 Bielefeld, Germany

{mkortkam, swachsmu}@TechFak.Uni-Bielefeld.DE

Abstract. The Growing Neural Gas (GNG) algorithm is able to perform continuous vector quantization for an unknown distribution while preserving the topological structure of the input space. This makes the GNG attractive for online learning of visual codebooks. However, mapping an input vector to a reference vector is quite expensive and requires an iteration through the entire codebook. We propose a hierarchical extension of the Growing Neural Gas algorithm for online one-shot learning of visual vocabularies. The method intrinsically supports mapping input vectors to codewords in sub-linear time. Further, our extension avoids overfitting and locally keeps track of the topology of the input space. The algorithm is evaluated on both, low dimensional simulated data and high dimensional real world data.

Keywords: online, constructive, one-shot, unsupervised, learning, visual, patterns, codewords, hierarchical, GNG, lbTreeGNG.

1 Introduction

In this work, we consider a *visual feature* as a description of either a local image patch or a global description of a complete image. Visual features are mathematically represented by vectors ξ in a high dimensional *feature space* $\mathcal{F} = \mathbb{R}^d$. An example for a local image description is the SURF feature [1]. A single SURF descriptor is a 128-dimensional vector that robustly represents the distribution of gradients in a local image patch, centered at an invariant key-point. Using SURF on an entire image leads to a set of (usually hundreds of) local visual features that represent the image. An example for a global image feature is the “tiny image” [2], which represents each color channel by a scaled version of the size 32×32 , resulting in a 3072-dimensional image description.

Any visual feature ξ that we may process is sampled from the unknown distribution $P(\xi)$, i.e. the overall distribution of all possible visual features in \mathcal{F} . We view a *visual pattern* as a local region in the feature space \mathcal{F} , which has a certain density w.r.t. $P(\xi)$. In common with the basic work of [3] we use the term *visual codewords* to refer to vectors in the feature space that approximate such local regions, e.g. in terms of clusters and centroids. Further, we also use

the term *visual codebook* to refer to a collection of visual prototype patterns, that emerge in \mathcal{F} under some distance measure in an unsupervised way [3].

For methods based on visual codebooks of local image descriptors - especially *bag of words* related models - sufficient object, category and scene recognition performances have been demonstrated. Some related examples are [3] [4] [5] [6] [7]. But also for global descriptors it has been shown on huge datasets that local neighborhoods in feature spaces can be used to achieve a reasonable performance in object recognition task [2].

To produce a visual codebook for local image descriptors, the majority of works pre-define the number of codewords and learn a vocabulary offline on a dataset, e.g. using k-means. After this preliminary step the methods just employ the static codebooks in the further processing. For those non-constructive offline methods, efficiency issues are well discussed in different works. For instance, in [4] a hierarchical k-means is used to partition the feature space. Besides, [6] explored the use of k-means with an approximate search.

However, in various situations it is neither applicable to pre-define the codebook size, nor to do offline learning. Instead, efficient constructive one-shot online learning methods are desirable. Some of the reasons for this type of learning can be:

- The amount of data to be processed is very high, similar to [2] [6].
- The resources are limited in time and space. For instance, this typically is the case for mobile robots like BIRON [8].
- Representative data is not given a priori. For instance, when learning a unknown home environment with a mobile robot from scratch [8].
- Input data is received in a stream-like way. For instance, when learning from Web-images related to text queries [7] [2] [5], we can treat the images gathered from the Web as a quasi-stream of images. Also data grabbed from cameras mounted on mobile robots [8] provide images in form of streams.
- Intermediate results of the learning process are needed, e.g. for the purpose of human robot interaction [5] [8].

In this paper, we contribute a novel efficient constructive one-shot online learning method to continuously learn and maintain visual codebooks. The approach is realized as an hierarchical extension of the *Growing Neural Gas* (GNG) [9] algorithm and called *limited branching tree Growing Neural Gas* (lbTreeGNG). Section 2 gives an overview of the basic idea. The approach intrinsically supports mapping input vectors to codewords in sub-linear time and avoids overfitting. Further, the learned structure locally sustains topological information of the feature space. The details of the algorithm can be found in section 3. We discuss the general behavior of the proposed method and provide results on both low dimensional simulated data (section 4) and high dimensional real world data (section 5).

2 The Basic Approach

To get in, consider figure 1. The first row shows a lbTreeGNG after processing 20,000 input signals uniformly sampled from the gray regions. Snapshots from

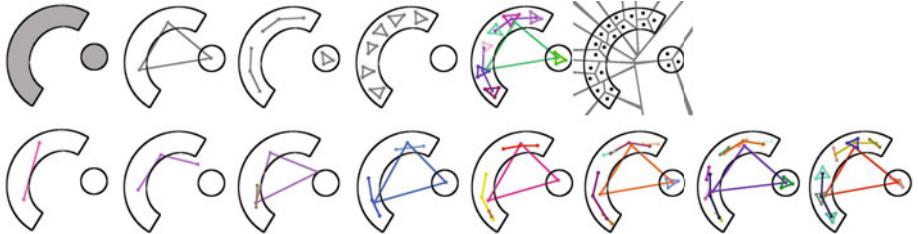


Fig. 1. (*First Row*): In this example, the input vectors for the lbTreeGNG are uniformly sampled from the gray regions. After training with 20,000 samples, the first three levels of the lbTreeGNG are shown subsequently. Then the overall lbTreeGNG is presented, followed by the hierarchical Voronoi-Tesselation induced by the learned tree. (*Second Row*): The second row shows the growth process of the lbTreeGNG during online learning. The corresponding iterations (from left to right) are 50, 125, 175, 250, 525, 750, 1975 and 3400. In the figure, colors of the GNGs are random. Further, points stand for nodes and lines between points stand for topological edges.

the process of growth can be seen in the second row. Given a maximum branching factor $b = 3$, the method starts with a single GNG in the first level of the tree. This GNG grows until it has b nodes. Subsequently, each of these nodes initializes a new local GNG in its Voronoi-Cell. Then the GNGs in the second level grow until they reach a size of b . The initialization and growing of GNGs in successive levels continue until a predefined node error m for the approximation is reached. Hence, GNGs in the third level are only created at positions that are not approximated well enough by the second level. The last two images in the first row show the overall lbTreeGNG and the hierarchical Voronoi-Tesselation induced by the codeword tree. In processing, the sampled input vectors are passed from the root GNG through the tree down to its nearest leaf node. Thereby, an input vector induces a major adaption in its corresponding leaf GNG and also propagates slight adaptions along the path up to the root GNG. We see that the hierarchical space tessellation directly supports an efficient mapping from input vectors to output vectors. In our simple example with 21 leaf nodes, the number of distance computations is $O(b \cdot \text{depth}) = 9$ in worst-case.

3 The lbTreeGNG Algorithm

In this chapter, we describe the lbTreeGNG algorithm and use a notation that is close to the original one of [9]. For a node x , we denote the associated reference vector with w_x . Note, that b and m are our newly introduced parameters. The parameter b limits the maximal branching factor of the learned codeword tree and thus controls the complexity of the method. The parameter m is the error threshold that effects the quantization error and guards the learner against overfitting. The original GNG parameters remain untouched and could be used with the following default values as proposed in [9]: $\epsilon_b = .2$, $\epsilon_n = .006$, $a_{max} = 50$,

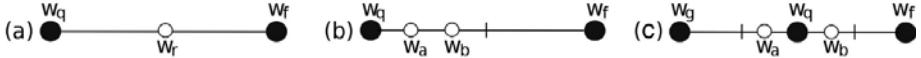


Fig. 2. Different cases in the node creation process. The variable q denotes the node with the highest error in a GNG, f is the neighbor node of q with the highest error and g the neighbor with the second highest error (if existing). Further, a and b are the initial nodes of a new child GNG that is inserted in the Voronoi-Cell of q . For any node x , w_x denotes the corresponding reference vector. (a) shows how a new node r is inserted in common GNGs. (b) and (c) visualize the two cases in which initial nodes a and b are created on q 's topological edges within the Voronoi-Cell of q .

$\alpha = .5$, $d = .995$ and $\lambda = 100$. All parameters in our method are used globally for all GNGs and do not change over time.

Main Loop. The lbTreeGNG is initialized with a single GNG. In the main loop of the algorithm, we generate an input ξ according to $P(\xi)$ and pass it from the root level GNG $l = 0$ to a leaf GNG on level $l = L$ following the nearest reference vectors. For each level $l = 0, \dots, L$ we keep track of the winner nodes $s_{1,l}$ and second winner nodes $s_{2,L}$. Thereafter, we adapt the winning leaf GNG of $s_{1,L}$ and the intermediate GNGs of $s_{1,l}$ with $l = 0, \dots, L - 1$. This strategy of passing vectors down a hierarchical tree, yields to very efficient (sub-linear) runtime complexity and is inspired from related work in computer vision [4] [5].

Adapt Winning Leaf GNG. Using $s_{1,L}$ and $s_{2,L}$ we do the same steps as the GNG algorithm [9], i.e. increment the edge age, increase the node error, move $s_{1,L}$ and $s_{2,L}$ towards the input, check the edge connectivity and the edge age. Then, every λ iteration we do some extended processing. First, to continue growth only in poorly approximated regions, we add a check whether the maximum node error of the leaf GNG is bigger than our parameter m . Only if this condition is true, we continue to refine our current codeword tree. The strategy to stop growing, is inspired by the work of [10]. Going on in the process, we reached a point where we want to create a new node and we discriminate the three basic situations explained below. Note, that we control the maximal branching factor of the resulting codeword tree to keep track of the worst-case runtime. Again, this is inspired from works using hierarchical k-means with a fixed branching factor in computer vision [4]. The explicit restriction of the branching factor to impose the worst-case complexity is also a major feature of our method in comparison to other hierarchical GCS [11] [12] and GNG [13] approaches. In the following, let q denote the node with the highest error in the GNG of $s_{1,L}$ and f the node with the second highest error f .

Create a New Node. If the GNG of $s_{1,L}$ has less than b nodes, we create a new node r in the common way [9] halfway between the node with the highest error q and the node with the second highest error f . This case is visualized in figure 2 (a). The corresponding equation is:

$$w_r = .5(w_q + w_f). \quad (1)$$

Create a New GNG. If the branching factor of the GNG is equal to the parameter b , we need to insert a new child GNG for the node with the highest error. For a new GNG, two initial nodes must be created. Like the creation of new nodes in common GNGs [9], we follow the principle of accumulated error minimization and place these initial nodes on the topological edges of q inside the Voronoi-Cell of q . Thereby, q has either one neighbor f or more neighbors. We let $g \neq f$ denote the neighbor node of q with the second highest error, if it is existing. The two different cases that arise can be seen in figure 2 (b) and (c). The corresponding equations are:

$$w_a = .875w_q + .125w_f \quad w_b = .625w_q + .375w_f \quad (2)$$

$$w_a = .750w_q + .250w_f \quad w_b = .750w_q + .250w_g. \quad (3)$$

Adapt Intermediate GNGs. Since the global tree should remain flexible at any time, it is necessary to adapt the GNGs of intermediate nodes as well. These nodes were traveled while passing the input vector down the tree and while walking through the nearest node in each GNG. We do the adaption in a winner-takes-all fashion and move all winner nodes $s_{1,l}$ of intermediate levels $l = 0, \dots, L - 1$ towards the input ξ :

$$\Delta w_{s_{1,l}} = \beta(\xi - w_{s_{1,l}}). \quad (4)$$

For a reasonable choice of β we must consider the following aspects. On the one hand, β should be sufficiently smaller than ϵ_b (the factor for the adaption of a winning node in a leaf GNG) to avoid pulling nodes out of their enclosing Voronoi-Cells in higher levels. On the other hand, the number of adaptions that are received in upper layers grow exponentially with b , e.g. in a balanced tree a node at level l can receive up to b^{L-l} updates. Hence, β should incorporate this by allowing gradually less movements in higher levels of the tree. Our choice of β takes both discussed aspects into account:

$$\beta = \epsilon_n b^{-(L-l+1)}. \quad (5)$$

4 Simulation

In this chapter we demonstrate the behavior of the algorithm on some artificial data and especially show the influence of the introduced parameters b and m .

Experiments. In this experiment we use the input distribution known from the example in figure 1. We use the default values for the GNG parameters as stated in section 3 and vary either b or m to demonstrate the resulting effects. The visual summary can be found in figure 3. The number of iterations is 20,000.

Discussion. In the left part of figure 1 we see the effect of altering the maximum branching factor b , while keeping all other parameters constant. If $b = INF$, then the algorithm only has one GNG in the root level. Smaller values like 2, 6, 10 induce different hierarchical Voronoi-Tesselations. We can see, that the

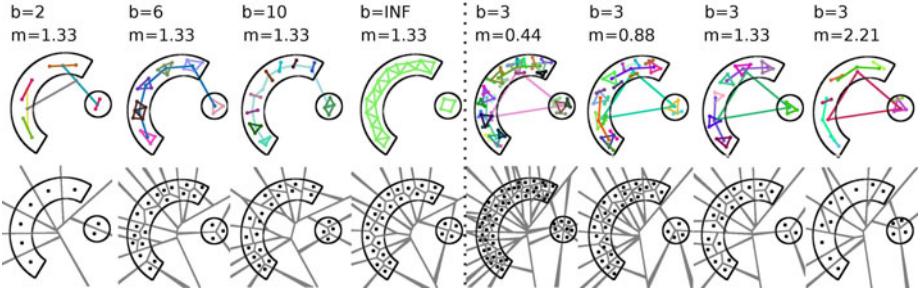


Fig. 3. On the new lbTreeGNG parameters. (*left:*) Shows the effect of changing the parameter b while keeping all other parameters constant. (*right:*) Shows the effect of changing the parameter m while keeping all other parameters constant.

parameter b effects the size of the local topologies and results in trees ranging from binary trees up to flat GNGs. The parameter is essential for the worst-case runtime complexity: Finding the nearest corresponding codeword for an input vector ξ takes at least $O(bL)$ distance calculations, where L is the maximum depth of the tree. The other parameter m controls the degree of approximation that is reached by the leaf nodes of the lbTreeGNG. The effects of varying m can be seen in the right part of Figure 11. A low error threshold leads to a large number of leaf nodes which realize a high approximation of the seen input data. A high error threshold leads to a rough approximation by few codewords.

5 Application

In this chapter we provide a set of experiments on high dimensional real world data to discuss different aspects of the proposed method.

5.1 Experiment I: Global Descriptors

In this experiment we aim at two things. First of all, we want to show that the effects of the parameters b and m (that was seen in chapter 4) also transfer to learning in high dimensional spaces. Second, we want to relate the outcome to a well known baseline method. In the experiment we employ pseudo-streams of global descriptors from the following well known datasets.

Datasets. We use the MNIST training data set [14] which contains 60,000 images of handwritten digits. For each gray-scale image of size $28 \times 28 \times 1$ we construct a column vector $v \in \mathbb{R}^{784}$ with normalized intensities in $[0, 1]$. Further, we use the 1.5 mil. tiny images data set provided by [2]. The data set contains $1.5 \cdot 10^6$ color images of size $32 \times 32 \times 3$. Roughly speaking, the images have been gathered from the Web by pumping a dictionary of English words into different image search engines. Thus, the tiny images contain a huge diversity of scenes

and objects. For each image we create a column vector in \mathbb{R}^{3072} with normalized intensities in $[0, 1]$.

Experiment. As in chapter 4, we use the default parameters for the GNGs and vary the values for parameter b and m respectively. We assess the structure and the quality of the hierarchical codebook in terms of different key numbers, e.g. the average pixel-wise reconstruction error on a data set given a learned codebook (`avRecErr`). Further, in order to provide a baseline for the reconstruction error we present results on the data sets using an offline learning k-means. For the overall view at the experiments and the results see table 1. The results in each column of the table are averaged over 10 runs of the algorithms and rounded appropriately.

Discussion. On the simulated data, we see that the maximum branching factor b directly effects the worst-case mapping runtime. For $b = INF$ the algorithm produces a flat non-hierarchical GNG. On the real data we can support our observations by the numerical values in table 1. We can see for both data sets that the worst-case runtime `wcRT` for mapping a vector to a leaf codeword grows as b goes up. The computational benefit from the hierarchical structure gets bigger when the codebooks are larger, especially in comparison to flat GNGs. In chapter 4, we show on the simulated data that the m parameter effects the approximation of the input distribution. On the real data sets we measure this in terms of `avRecErr` and `avRecErrVar` in table 1. Given a constant b , we can see for both data sets that `avRecErr` and `avRecErrVar` increase as m gets larger. The performance in terms of `avRecErr` and `avRecErrVar` is comparable to k-means. The difference between `lbTreeGNG` and k-means reconstruction error is usually not large in terms of image gray-scale values, e.g. less than a single gray-scale value per pixel for the first column of the table. Overall, the results show that the introduced parameters can be used to control the underlying trade-off between speed and accuracy, while maintaining reasonable codebooks. Figure 4 shows some exemplarily visual codewords.

5.2 Experiment II: Local Descriptors

In this experiments we want to assess the behavior of the learner on local image descriptors over time.

Dataset. We merged three different Web-image datasets together [15] [7] [16], resulting in a collection of approx. 90,000 images. The images were scaled to approx. 100,000 pixels.

Experiments. From the overall image collection, we use 75,000 images as a training set and sample 15,000 images as a test set. Totally, a pseudo-stream of approx. 44 million SURF [4] descriptors is processed while learning. During training, we repeatedly measure the reconstruction error on a the test set and the current number of leaf nodes. The results are averaged over multiple runs.

Discussion. In the plot in figure 4 we see the characteristic progress of the test reconstruction error and the number of leaf nodes during learning on a

Table 1. Results on real data sets. **avRecErr**: average reconstruction error for a pixel; **avRecErrVar**: variance of avRecErr; **numOfLeafs**: number of leafs/ codebook size; **maxDepth**: maximum depth of codebook tree; **wcRT**: worst-case runtime for mapping a vector to a leaf codeword $O(\maxDepth \cdot b)$; **numOfGNGs**: number of GNGs in the lbTreeGNG; **avBrachFac**: average branching factor of the codebook tree.

lbTreeGNG												
Data	MNIST						1.5 Mil. Tiny Images					
<i>b</i>	3	6	INF	3	3	3	3	6	INF	3	3	
<i>m</i>	10	10	10	30	50	70	150	150	150	70	80	120
avRecErr	.0066	.0065	.0062	.0069	.0076	.0084	.0031	.0031	.0030	.0030	.0030	.0031
avRecErrVar	.0022	.0021	.0015	.0019	.0015	.0014	.0021	.0021	.0021	.0020	.0020	.0021
numOfLeafs	478	505	602	316	120	26	2769	2802	3066	9509	8570	4673
maxDepth	12	22	1	12	9	7	18	28	1	19	18	17
wcRT	36	132	602	36	27	21	54	168	3066	57	54	51
numOfGNGs	297	180	1	193	72	15	1714	1004	1	5817	5286	2866
avBrachFac	2.6	3.8	602	2.6	2.7	2.7	2.6	3.8	3066	2.6	2.6	2.6
<i>k</i>	478	505	602	316	120	26	2769	2802	3066	9509	8750	4673
avRecErr	.006	.006	.0059	.0062	.0065	.0073	.0029	.0029	.0029	.0028	.0029	.0029
avRecErrVar	.0017	.0018	.0017	.0018	.0017	.0015	.0021	.0021	.0021	.0020	.0020	.0020
k-means												

stream of $4.4 \cdot 10^7$ SURF descriptors. In the beginning, by adding few nodes the learner yields a large drop down of the reconstruction error. After that, on a trained lbTreeGNG much more nodes need to be added to the codebook to realize relatively small error reductions. Finally, we see a saturation of the growth and the reconstruction error. Figure 4 exemplarily presents some visual codewords.

6 Other Related Work

In this section we discuss other related work, that was not already mentioned. The ground work for our approach is the Growing Neural Gas [9]. Various extensions of this algorithm exist and are related to our approach, e.g. [17] [10] [18] [13]. In [17] an utility criteria is introduced and is used to remove “useless” nodes and to make the GNG able to track non-stationary distributions. Note that this principle can also be applied to our method. The work of [10] suggests an alternative insertion mechanism, where new nodes are added whenever the current input is not sufficiently matched by the network. Further, [18] propose a hierarchical overlapping growing neural gas to perform automatic video shot motion characterization. The authors combine supervised and unsupervised learning in the GNG. Another hierarchical GNG extension is presented in [13]. The algorithm maintains a time history to be able to correct poor decisions made in the construction. Further, it allows to influence the shape and form

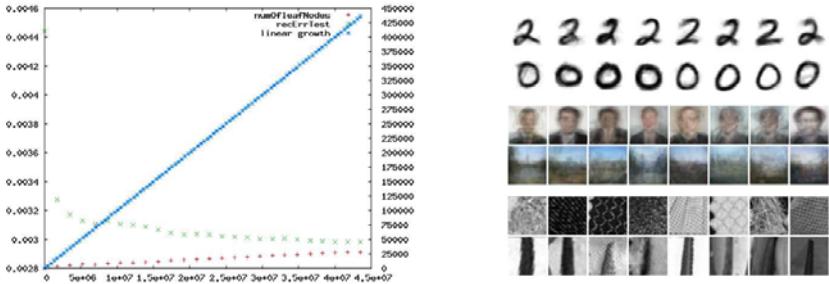


Fig. 4. (left-side) The left plot shows the characteristic progress of the test reconstruction error (green) and the number of leaf nodes (red) during learning on a stream of approx. $4.4 \cdot 10^7$ SURF descriptors. (right-side) The right side exemplarily shows visualisations of some topologically neighbored leaf codewords on the different datasets: MNIST (top), Tiny Images (middle), merged SURF (bottom).

of the learned hierarchy, but not as directly as with our limited branching restriction. Consequential, the authors implement depth and breadth in a more open way.

A similar approach to the GNG for unsupervised topology learning is called Growing Cell Structures (GCS) [19]. There are also several extended versions related to our method, e.g. [11] [12] [20]. In [11] an hierarchical GCS is presented. The authors do not change the weights of upper layers after subtrees are spanned. Further, the size of each layer is not bounded and it is - along with the number of layers and the node deletion frequency - dynamically determined. The work of [12] presents a hierarchical GCS version that is more robust against the ordering of the input vectors. In [20], the authors present a special approach of learning cell-structures that focuses on life-long learning. The underlying idea and motivations of those hierarchical schemes are quite similar to ours. However, due to the fact that those methods build upon another algorithm, there are basic differences in the implementation.

Besides, there are interesting related works on incremental and hierarchical clustering that employ related techniques and follow similar goals, i.e. [21] [22].

7 Conclusions

We presented an hierarchical extension of the GNG [9] for continuous learning of visual codebooks. The method has two additional parameters b and m and preserved the original GNG parameters. We discussed the influence of the parameters and the behavior of the algorithm on both low dimensional simulated and high dimensional real world data. Overall, we showed that our method locally preserves the topological structure of the input space and allow efficient classification of novel input signals.

References

1. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
2. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large database for non-parametric object and scene recognition. *Pattern Analysis and Machine Intelligence* 30(11), 1958–1970 (2008)
3. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV (2003)
4. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2006)
5. Collins, B., Deng, J., Kai, L., Fei-Fei, L.: Towards scalable dataset construction: An active learning approach. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 86–98. Springer, Heidelberg (2008)
6. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007)
7. Schroff, G.F., Criminisi, A., Zisserman, A.: Harvesting image databases from the web. In: ICCV (2007)
8. Peltason, J., Siepmann, F.H.K., Spexard, T.P., Wrede, B., Hanheide, M., Topp, E.A.: Mixed-initiative in human augmented mapping. In: ICRA (2009)
9. Fritzke, B.: A growing neural gas network learns topologies. In: NIPS, pp. 625–632 (1995)
10. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organising network that grows when required. *Neural Networks* 25(8-9), 1041–1058 (2002)
11. Burzevski, V., Mohan, C.K.: Hierarchical growing cell structures. In: Neural Networks (1996)
12. Hodge, V.J., Austin, J.: Hierarchical growing cell structures: TreeGCS. *Knowledge and Data Engineering* 13(2), 207–218 (2001)
13. Doherty, K.A.J., Adams, R.G., Davey, N.: Hierarchical growing neural gas. In: Adaptive and Natural Computing Algorithms, pp. 140–143 (2005)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11), 2278–2324 (1998)
15. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge (VOC 2009) Results (2009),
<http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html>
16. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)
17. Fritzke, B.: A self-organizing network that can follow non-stationary distributions. In: Artificial Neural Networks (1997)
18. Cao, X., Suganthan, P.N.: Video shot motion characterization based on hierarchical overlapped growing neural gas networks. *Multimedia Systems* 9(4), 378–385 (2003)
19. Fritzke, B.: Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks* 7(9), 1441–1460 (1994)
20. Hamker, F.H.: Life-long learning cell structures – continuously learning without catastrophic interference. *Neural Networks* 14(4-5), 551–573 (2001)
21. Campos, M.M., Carpente, G.A.: S-TREE: Self-organizing trees for data clustering and online vector quantization. *Neural Networks* 14(4), 505–525 (2001)
22. Furao, S., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. *Neural Networks* 19(1), 90–106 (2006)

An Efficient Collaborative Recommender System Based on k -Separability

Georgios Alexandridis, Georgios Siolas, and Andreas Stafylopatis

Department of Electrical and Computer Engineering,
National Technical University of Athens,
Zografou 15780, Athens, Greece

{gealexandri,gsiolas}@islab.ntua.gr, andreas@cs.ntua.gr

Abstract. Most recommender systems usually have too many items to recommend to too many users using limited information. This problem is formally known as the *sparsity* of the ratings' matrix, because this is the structure that holds user preferences. This article outlines a collaborative recommender system, that tries to amend this situation. The system is built around the notion of k -separability combined with a constructive neural network algorithm.

Keywords: collaborative recommender, sparsity problem, k -separability, constructive ANN architecture.

1 Introduction

The digital revolution of the past decades has led to the exponential growth of the information available in electronic form. The convergence of Telecommunications with Computing, the advent of the Internet and the World-Wide Web has also resulted in the fast diffusion of data in all forms of computing devices (*i.e* Desktops, Laptops, Personal Digital Assistants, Smartphones). This situation has brought about convenience, but it has also contributed to a problem known as the *information overload*. Nowadays, it has become quite common for users to be confronted with too many options when they are searching for something or when they are trying to make a decision. Therefore, the need for systems that will *index*, *search* and *categorize* information is becoming more and more evident in the course of time. *Recommender Systems* have been proposed as a solution to this problem. Their purpose is to filter the available information and present its most *relevant* and *interesting* instances. Their primal use is to propose new items that would be of interest to the users. The suggested items can be of almost any nature such as *books* (Amazon.com [1]), *movies* (MovieLens [2]), *music* (last.fm [3], *Pandora* [4]) or *news* stories. This paper introduces a new Recommender System, based on a neural network architecture, that tries to overcome one of the problems faced by modern-day Recommender Systems; that of the sparsity of the contained information (Section 3). A detailed overview of the underlying methodology (k -Separability) is given in Section 4, while the neural network details are discussed in Section 5. The whole system is described in Section 6, while in Section 7 experimental results are presented and analyzed.

2 Recommender Systems

Most Recommender Systems make use of the *ratings* of items as an indication of how much a particular user liked a particular item. Ratings are provided by the users themselves and comprise of discrete values in a given scale. They may be binary (i.e. *like/dislike*) or they may belong to a broader set, as depicted in Table I. In this example, users rate items in a five-degree discrete scale (1: strong dislike, 5: strong like), while the empty cells denote items not yet rated by the users. The role of the recommender system is to try to predict the “missing” values; that is, to try to “guess” the rating a particular user would have given to a particular item (in our example, how would the first user have rated the fourth item or how the second user would have rated the third item, etc). Once the prediction mechanism produces results, a trivial rule can be used to make the actual recommendations (i.e. propose the first n items with the biggest score). Recommender Systems may be categorized according to the way they make predictions. Most of the systems fall in one of the three categories described below:

- *Content-Based* Recommender Systems, where the new items that are presented to a specific user share some common characteristics with the items this user liked in the past.
- *Collaborative* Recommender Systems, where the new items that are presented to a specific user are those that users with similar taste have liked in the past.
- *Hybrid* Recommender Systems, where the two above-mentioned techniques are used in conjunction.

Content-Based Recommender Systems require additional information about the items before they are able to produce meaningful recommendations. On the contrary, Collaborative Recommender Systems do not necessarily use extra information about the users; they may try to locate similarities (or dissimilarities) by examining the ratings’ matrix. In the example illustrated by Table I, U_1 and U_2 seem to have dissimilar taste; and since U_1 dislikes I_3 , it would be logical for our system to assume that U_2 likes it. Likewise, U_2 and U_4 seem to have a common taste and therefore it would also be valid to assume that U_4 would “like” I_2 .

Table 1. Example Ratings’ Matrix of a Recommender System

	I_1	I_2	I_3	I_4
U_1	5	3	2	
U_2	3	5		2
U_3	1		2	
U_4	2			3

3 Challenges in Recommender System Design

Recommender Systems have been implemented using a variety of methodologies, like *neural networks*, *genetic algorithms*, *Bayesian classifiers*, etc [2], [5]. All implementations so far have suffered from two fundamental problems: the *cold-start* problem and the *sparsity* of the ratings matrix.

3.1 The Cold-Start Problem

The cold-start problem refers to the condition when a new user is entered into the system. Since this new user has not made any ratings yet, it is difficult for the recommender system to make a meaningful recommendation. However, this problem is more related to the underlying philosophy of the system than to the recommender algorithm itself. Indeed, when no information is available, even the best recommender algorithm will have to “guess” rather than make predictions. This problem may be amended in several ways (e.g. recommend the most popular items or ask the new user to make a few ratings prior to making any recommendation) that are irrelevant to the performance of the recommendation algorithm itself.

3.2 The Sparsity Problem

On the other hand, the *sparsity* of the ratings’ matrix can be a major drawback of the recommender algorithm. In most real case applications, the users of the system and the items to be recommended continuously increase; therefore the density of the matrix will remain very sparse (only about 5-10% of its elements will be non-zero), even though more and more users rate more and more items. It is obvious that little or no knowledge could be extracted from a large and mostly empty matrix. As a first preprocessing step, *dimensionality reduction* techniques may be employed, which lower the size of the matrix, while at the same time maintaining its semantic structure, incurring the smallest information loss possible. Many techniques can be used to reduce the dimensionality of multi-dimensional data, like *Principal Component Analysis* (PCA), *Linear Discriminant Analysis* (LDA) and others. One approach that yields good results [10] is *Latent Semantic Indexing* which is based upon the *Singular Value Decomposition* (SVD). If an $r \times c$ matrix A is sparse, then its rank m is substantially smaller than the number of its rows ($m \ll r$), and a special case of the SVD may be used, the *compact SVD*:

$$A = U_m \Sigma_m V_m^T \quad (1)$$

The initial matrix A is recalculated as the product of three other matrices, U, Σ and V , where U and V contain the *left* and *right* singular vectors of the initial matrix A . Σ is a diagonal matrix whose non-zero elements are the singular values of A , appearing in decreasing order. The singular values quantify the amount of variance in the original data. By keeping the m non-zero singular values along with their corresponding singular vectors, the latent structure of the initial matrix is maintained to a satisfying degree.

4 k -Separability

Multi-layered neural networks [6] function as universal approximators by using their hidden layers to transform their input into linearly separated data clusters. Although this procedure yields significant results in many problems, it performs below average on datasets that are not linearly separable. The most common example is the XOR Problem (and its generalization, the parity problem) but it could be easily argued that most real-life datasets are inherently linearly inseparable.

A variety of methods have been proposed to overcome linear inseparability. Support Vector Machines (SVMs) [6] propose the projection of the input data into a higher-dimensional space, where it would be easier to locate a hyperplane that would separate the data clusters. Although SVMs provide good generalization performance, calculating the support vector is a computationally intensive task in many cases. Radial-Basis Functions (RBFs) work in a similar way [6], by projecting input data into a higher dimensional space, but this time the objective is to find a surface that provides a best fit for the input data with the criterion defined in some statistical sense.

k -Separability was proposed by Duch [7], [8] and Grochowski [8] and is a special application of a more general method that is called *projection pursuit* [3]. *Projection pursuit* is a statistical technique that involves finding the most “interesting” possible projections of multidimensional data in lower dimensions, therefore reducing the overall computational complexity of the problem at hand. “Interesting” projections (in a statistical sense) are applied to data and as a result their dimension is reduced. This process is iteratively repeated (the “pursuit” aspect) in order to discover new projections that would allow for a further projection to an even lower-dimensional space.

k -Separability tries to extend the notion of linear separability of data into $k \geq 2$ segments on the discriminating hyperplane. This is better illustrated in Figure 11 for the 2-bit XOR Problem. The activation function of the output neuron partitions the input space into more than two distinct areas (Fig. 11). Good candidate functions for this task are the soft-windowed functions that are derived from the multiplication of two sigmoidal functions that have a phase difference [8]:

$$M(\mathbf{x}; \mathbf{w}, a, b, \beta) = \frac{1}{2}(1 - \tanh(\beta(\mathbf{w}\mathbf{x} - a))\tanh(\beta(\mathbf{w}\mathbf{x} - b))) \quad (2)$$

where \mathbf{w} denotes the *weight* vector, \mathbf{x} the input vector and β controls the slope of the the sigmoidal functions (producing a hard-windowed transfer function when set to large values). a and b are the extra bias parameters that act as the boundary for the different regions (Fig. 11). It should be noted that M is a *squashing* function (it converges to a fixed value when its input reaches positive or negative infinity) [6] that is differentiable; therefore it is fit for use as an activation function of a neuron. It is also trivial to prove that, when the *induced local field* $\mathbf{w}\mathbf{x}$ falls in the $[a, b]$ interval, M is positive and zero otherwise. If the slope β is set to a high value, then $M = 1$ when $\mathbf{w}\mathbf{x} \in [a, b]$. The extra

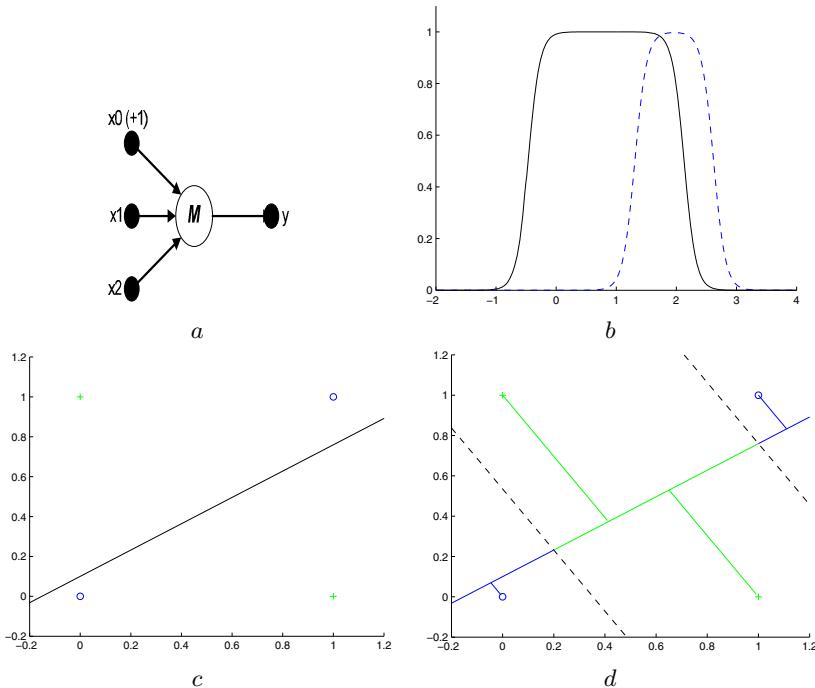


Fig. 1. 2-bit XOR Problem. (a) Neural Network Architecture (b) Activation Function: before training (dashed line) and after (continuous line) (c) Data points and the discriminating hyperplane (line) (d) Projection of the data points onto the discriminating line and the formation of the 3 regions.

bias parameters are adjusted during training and therefore the “shape” of the transfer function changes as training proceeds, according to the characteristics of the input. This is better illustrated in Fig. 1(b) where the dotted line depicts the transfer function before training and the continuous line depicts the same function after training is over.

Training can take place using any back-propagation algorithm; in our experiments we used the *gradient descent with momentum and adaptive learning rate backpropagation* method [6] because it appeared to have better results in convergence. The initial values for the extra bias parameters can be set to [8]:

$$\begin{aligned} a &= (\mathbf{w}\mathbf{x})_{min} + \frac{1}{3}|(\mathbf{w}\mathbf{x})_{max} - (\mathbf{w}\mathbf{x})_{min}| \\ b &= (\mathbf{w}\mathbf{x})_{min} + \frac{2}{3}|(\mathbf{w}\mathbf{x})_{max} - (\mathbf{w}\mathbf{x})_{min}| \end{aligned} \quad (3)$$

and they can be updated like the bias.

The architecture presented so far extends the linear separability property of a single neuron to only one level up, implementing a 2-separable decision boundary. It is ideal for decision problems that are described by datasets that

are clustered in 3 different regions when projected to a single dimension (line). The network training in this concept is about finding the discriminating line and the boundaries set by the extra bias parameters a and b ($(-\infty, a]$, $(a, b]$ and (b, ∞)) (Fig. ⑩).

5 Neural Network Architecture

Datasets which form more than 3 clusters when projected on the discriminating line may be learned by the combination of the output of two or more neurons. For example, a 5-separable dataset may be learned by combining the output of two neurons. In that case, the discriminating line is divided into five distinct regions. If the extra bias parameters of the two neurons are set properly, then at most one neuron will “fire” at any given input sample, indicating the cluster to which the sample belongs.

Extending the reasoning described above, the combined output of m neurons leads to the formation of $k = 2m + 1$ distinct regions on the discriminating line. This network may be implemented by a Multi-Layer Perceptron architecture with one hidden layer and one output layer (Fig. ②). The n -dimensional network input is projected onto the m neurons, with window-type activation functions, which form the $2m + 1$ regions in the discriminating line. The outputs of the hidden-layer neurons are then given as input to the output layer, which functions as an adder, and the total network output is computed. This network implements an $n \rightarrow m \rightarrow 1$ projection and is appropriate for a recommendation task; given an input item described by n characteristics (in content-based recommenders) or the ratings on the item by the n most “like minded”-users (collaborative recommender) the network can decide whether this item will be liked by the user or not.

5.1 Network Construction

Network Construction could have been trivial had the number of discrete clusters of the input data been known *a priori*; in that case, the x clustered input data

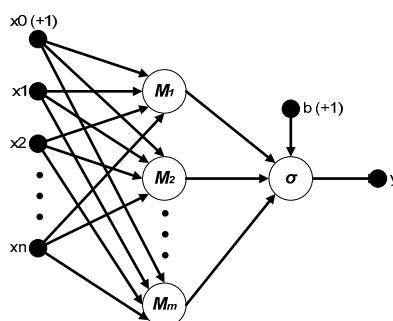


Fig. 2. MLP Architecture. n input nodes, m neurons in the hidden layer (window-type activation function), one output layer (sigmoidal function)

could be learned by a 2-layered perceptron with $m \approx \frac{x-1}{2}$ neurons in the hidden layer and one output neuron (Figure 2). Unfortunately, this is not the case. The number of clusters cannot be estimated until training has been well advanced and the proper discriminating line and boundaries have been found. Setting x to a constant value is of little help because it may either be smaller than k , in which case the network training error will be high, or larger than k , in which case the the resulting network will be overfitted.

Therefore, k may be estimated by employing either *constructive* network architecture algorithms, that start with a minimal network topology and add neurons as training progresses, or *pruning* network architecture algorithms, that firstly train a full network architecture and then remove the neurons that contribute to the network output the least. Selecting the appropriate algorithm requires some knowledge of the problem at hand. In the case of collaborative recommender systems with sparse ratings' matrices as their input, the overall network size is expected to be small. Therefore, it is reasonable to select a constructive network algorithm. In our approach we have adopted the algorithm presented in [4] in conjunction with AdaBoost [9], an algorithm for selecting specific samples from the training set based on the training error.

The Algorithm works as follows. Initially, a minimal ANN architecture is created, with only one neuron in the hidden layer. This network is trained in two phases (initial and final partial training) on the whole training dataset, using the back-propagation algorithm discussed above. When training finishes and the classification error on each sample has been computed, the AdaBoost algorithm is employed. A new training set is created (with samples taken from the original dataset) where the samples with the biggest classification error are likely to appear more frequently. A new neuron is added in the hidden layer and the training procedure is repeated (again, in two phases) but this time only the new neuron's weights and biases are adapted, while all other network weight and biases remain "frozen". As a result, adding new neurons in the hidden layer is equivalent to trying to "learn" the yet unlearned samples and to discover new clusters. The network construction stops when the new neurons added in the hidden layer have little impact on the classification error.

6 Collaborative Recommender System

Based on the concepts discussed above, the proposed Collaborative Recommender System takes as input the ratings' matrix M of all users and outputs a model (a neural network) for each user, in the following steps:

1. Pick from the *ratings matrix* M all the users (rows) who have rated at least two items in common with the *target user* and create matrix A .
2. Compute the SVD of the $r \times c$ matrix A of rank m . Since A is a sparse matrix, $m \ll r$
3. Calculate matrix A' by maintaining all the m non-zero singular values of A . A' is an $n \times n$ matrix of much lower dimensionality than the initial matrix A (in our case, $n = m$).

4. Partition the matrix A' in 3 different sets; the *training* set, the *validation* set and the *test* set, in a 60%-20%-20% ratio.
5. Using the training and validation sets, train an ANN with the constructive network algorithm described above, along with the AdaBoost method.
6. After the network training is over, compute the performance metrics on the test set.

7 Experimental Results and Conclusion

The proposed Collaborative Recommender System has been tested on the *MovieLens Database* [12]. The MovieLens database contains the ratings of 943 users on 1682 movies. Each user has rated at least 20 movies on a 5 degree discrete scale (1 denotes strong dislike, 5 denotes strong like). The ratings' matrix contains 100.000 entries, which means that it is to a large extent a sparse matrix (only 6.3% of its total elements are non-zero). Even though the top user has rated 737 out of 943 movies, the *rated items per user* variable follows an exponential distribution as shown in Figure 3. Almost 60% of the users have rated 100 movies or less and another 40% of the users have rated 50 movies or less.

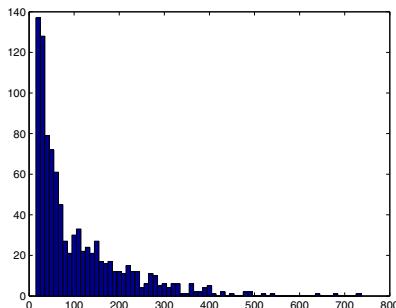


Fig. 3. Probability distribution of the *rated items per user* variable in the *MovieLens Database*

The MovieLens database provides additional information about each user (i.e age, gender, occupation) and movie (i.e title, genre, release date) but our Collaborative Recommender System made exclusive use of the ratings' matrix. That is, the only information considered about each user and movie was their respective id. The ratings were further discretized, by mapping them onto two classes; the “-1” class for *dislike* (ratings 1 to 3) and the “+1” class for *like* (ratings 4 to 5).

We considered two distinct user groups. *Group A* contained all users who have rated 20-50 movies (the few raters user group), while *Group B* contained all users who have rated 51-100 movies (the moderate raters user group). We randomly picked 20 users from each group (40 users in total) and created a model for each one of them. The results were averaged for each of the two groups.

Table 2. Performance Results

Methodology	Precision	Recall	F-measure
OurSystem: User Group A (few ratings)	74.07%	88.86%	78.97%
OurSystem: User Group B (moderate ratings)	75.38%	82.21%	79.37%
Movielens	66%	74%	70%
MovieMagician Feature-based	61%	75%	67%
MovieMagician Clique-based	74%	73%	74%
MovieMagician Hybrid	73%	56%	63%
Correlation	64.4%	46.8%	54.2%
SVD/ANN	67.9%	69.7%	68.8%

In the literature, the comparison of the various Recommender Systems is based upon two sets of metrics; the *Mean Average Error* (MAE) and the *Precision-Recall* metric. The MAE measures the difference between the output of the Recommender System versus its desired output. *Precision* is the percentage of the movies recommended that were actually liked by the target user, while *Recall* is the percentage of the movies recommended to the total number of the liked movies. The *F-measure* is a quantity defined as [10]:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

In our experiments, we used the latter set of metrics as they are more appropriate for the recommendation environment, where what is of interest is the quality of the recommended items and not how close their output is to the actual score given by each user. Our system's results for the two groups of users are presented in Table 2 and are compared to those of MovieMagician [11] (next 4 lines of Table 2) and of some earlier implementations [10] (last 2 lines of Table 2). The results indicate that the proposed method outperforms the remaining approaches.

Our Collaborative Recommender System seems to be robust as it achieves good results in both scenarios (users with few and moderate ratings). A first observation is that for Group A *Recall* is higher compared to Group B. The reason is that many users with few ratings tend to rate only the movies they like and therefore any movie recommended (out of their respective *test set*) will most probably be liked by them. Nevertheless, our system accomplishes a good trade-off between *Precision* and *Recall*, a basic requirement for all Recommender Systems. This is due to the fact that k -separability is able to uncover more complex statistical dependencies (both positive and negative) compared to other methods. Because of that, our algorithm does not need to further process the neighborhood of “similar” users by using methods such as the *Pearson Corellation* formula. In that sense, all the neighbors of the selected user are taken into consideration, something which is extremely useful in those cases where both the ratings and the co-rates are few, a situation very common in almost every recommender.

References

1. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7(1), 76–80 (2003)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17(6), 734–749 (2005)
3. Friedman, J.H., Tukey, J.W.: A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computers C-23* (9), 881–890 (1974), ISSN 0018-9340
4. Islam, M.M., et al.: A new constructive algorithm for architectural and functional adaptation of artificial neural networks. *IEEE Trans. Sys. Man Cyber. Part B* 39(6), 1590–1605 (2009)
5. Christakou, C., Vrettos, S., Stafylopatis, A.: A Hybrid Movie Recommender System Based on Neural Networks. *International Journal on Artificial Intelligence Tools* 16(5), 771–792 (2007)
6. Haykin, S.: Neural Networks and Learning Machines, 3rd edn., 936 Pages. Prentice-Hall, Englewood Cliffs (2008), ISBN 978-0131471399
7. Duch, W.: k -Separability. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 188–197. Springer, Heidelberg (2006)
8. Grochowski, M., Duch, W.: Constructive Neural Network Algorithms that Solve Highly Non-Separable Problems. In: Constructive Neural Networks. Springer Studies in Computational Intelligence, vol. 258, pp. 49–70 (2010)
9. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning, Proceedings of the Thirteenth International Conference pp. 148–156 (1996)
10. Billsus, D., Pazzani, M.J.: Learning Collaborative Information Filters. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 46–54 (1998)
11. Grant, S., McCalla, G.: A hybrid approach to making recommendation and its application to the movie domain. In: Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence, pp. 257–266 (2001)
12. MovieLens, GroupLens Research, University of Minnesota,
<http://movielens.umn.edu/>
13. last.fm, <http://www.last.fm/>
14. Pandora, <http://www.pandora.com/>

Empirical Analysis of the Divergence of Gibbs Sampling Based Learning Algorithms for Restricted Boltzmann Machines

Asja Fischer and Christian Igel

Institut für Neuroinformatik
Ruhr-Universität Bochum, 44780 Bochum, Germany
{asja.fischer,christian.igel}@ini.rub.de

Abstract. Learning algorithms relying on Gibbs sampling based stochastic approximations of the log-likelihood gradient have become a common way to train Restricted Boltzmann Machines (RBMs). We study three of these methods, Contrastive Divergence (CD) and its refined variants Persistent CD (PCD) and Fast PCD (FPCD). As the approximations are biased, the maximum of the log-likelihood is not necessarily obtained. Recently, it has been shown that CD, PCD, and FPCD can even lead to a steady decrease of the log-likelihood during learning. Taking artificial data sets from the literature we study these divergence effects in more detail. Our results indicate that the log-likelihood seems to diverge especially if the target distribution is difficult to learn for the RBM. The decrease of the likelihood can not be detected by an increase of the reconstruction error, which has been proposed as a stopping criterion for CD learning. Weight-decay with a carefully chosen weight-decay-parameter can prevent divergence.

Keywords: Unsupervised Learning, Restricted Boltzmann Machines, Contrastive Divergence, Gibbs Sampling.

1 Introduction

Training large undirected graphical models by vanilla likelihood maximization is in general computationally intractable because it involves averages over an exponential number of terms. Obtaining unbiased estimates of these averages by Markov chain Monte Carlo methods typically requires many sampling steps. However, biased estimates obtained after running a Gibbs chain for just a few steps can be sufficient for model training [1]. This is exploited by *Contrastive Divergence* (CD, [1]) learning and its variants *Persistent CD* (PCD, [2]) and *Fast PCD* (FPCD, [3]), which have been, for example, successfully applied to training of *Restricted Boltzmann Machines* (RBMs), the building blocks of *Deep Belief Networks* (DBNs) [4,5].

Contrastive Divergence learning is a biased approximation of gradient-ascent on the log-likelihood of the model parameters and thus does not necessarily reach the maximum likelihood estimate of the parameters. The bias depends on

the mixing rate of the Markov chain, and mixing slows down with increasing model parameters [167]. Recently it has been shown that the bias can lead to a divergence of the log-likelihood when training RBMs [39]. In this study, we further investigate this divergence behavior and its dependence on the number of sampling steps used for the approximation, the number of hidden neurons of the RBM, and the choice of the weight decay parameter. After a brief description of CD, PCD, and FPCD, we describe our experiments, discuss the results and finally draw our conclusions.

2 Training RBMs

An RBM is an undirected graphical model [10]. Its structure is a bipartite graph consisting of one layer of visible units $\mathbf{V} = (V_1, \dots, V_m)$ to represent observable data and one layer of hidden units $\mathbf{H} = (H_1, \dots, H_n)$ to capture dependencies between observed variables. It is parametrized by the connection weights w_{ij} as well as the biases b_j and c_i of visible and hidden units, respectively ($i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$). Given these parameters, jointly denoted as $\boldsymbol{\theta}$, the modeled joint distribution of \mathbf{V} and \mathbf{H} is $p(\mathbf{v}, \mathbf{h}) = e^{-\mathcal{E}(\mathbf{v}, \mathbf{h})}/Z$, where $Z = \sum_{\mathbf{v}, \mathbf{h}} e^{-\mathcal{E}(\mathbf{v}, \mathbf{h})}$ and the energy \mathcal{E} is given by

$$\mathcal{E}(\mathbf{v}, \mathbf{h}) = - \sum_{i=1}^n \sum_{j=1}^m w_{ij} h_i v_j - \sum_{j=1}^m b_j v_j - \sum_{i=1}^n c_i h_i .$$

Differentiating the log-likelihood $\ell(\boldsymbol{\theta} | \mathbf{v}_l)$ of the model parameters $\boldsymbol{\theta}$ given one training example \mathbf{v}_l with respect to $\boldsymbol{\theta}$ yields

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ell(\boldsymbol{\theta} | \mathbf{v}_l) = - \sum_{\mathbf{h}} p(\mathbf{h} | \mathbf{v}_l) \frac{\partial \mathcal{E}(\mathbf{v}_l, \mathbf{h})}{\partial \boldsymbol{\theta}} + \sum_{\mathbf{v}} p(\mathbf{v}) \sum_{\mathbf{h}} p(\mathbf{h} | \mathbf{v}) \frac{\partial \mathcal{E}(\mathbf{v}, \mathbf{h})}{\partial \boldsymbol{\theta}} . \quad (1)$$

Computing the first term on the right side of the equation is straightforward because it factorizes. The computation of the second term is intractable for regular sized RBMs because its complexity is exponential in the size of the smallest layer. However, the expectation over $p(\mathbf{v})$ can be approximated by alternating Gibbs sampling [1112]. But since the sampling chain needs to be long to get almost unbiased samples of the distribution modeled by the RBM, the computational effort is still too large.

Contrastive divergence. Instead of running the Gibbs chain until a near-to-equilibrium distribution is reached, in the k -step Contrastive Divergence (CD $_k$) algorithm [1] the chain is run for only k steps, starting from an example $\mathbf{v}^{(0)}$ of the training set and yielding the sample $\mathbf{v}^{(k)}$. Each step t consists of sampling $\mathbf{h}^{(t)}$ from $p(\mathbf{h} | \mathbf{v}^{(t)})$ and sampling $\mathbf{v}^{(t+1)}$ from $p(\mathbf{v} | \mathbf{h}^{(t)})$ subsequently. The gradient [1] with respect to $\boldsymbol{\theta}$ of the log-likelihood for one training pattern $\mathbf{v}^{(0)}$ is then approximated by

¹ When referring to sizes of model parameters, we refer to their absolute values.

$$\text{CD}_k(\boldsymbol{\theta}, \mathbf{v}^{(0)}) = - \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}^{(0)}) \frac{\partial \mathcal{E}(\mathbf{v}^{(0)}, \mathbf{h})}{\partial \boldsymbol{\theta}} + \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}^{(k)}) \frac{\partial \mathcal{E}(\mathbf{v}^{(k)}, \mathbf{h})}{\partial \boldsymbol{\theta}} . \quad (2)$$

In the following, we restrict our considerations to RBMs with binary units for which $E_{p(h_i|\mathbf{v})}[h_i] = \text{sigmoid}\left(c_i + \sum_{j=1}^m w_{ij} v_j\right)$ with $\text{sigmoid}(x) = (1 + \exp(-x))^{-1}$.

The expectation $E_{p(\mathbf{v}^{(k)}|\mathbf{v}^{(0)})} [\text{CD}_k(\boldsymbol{\theta}, \mathbf{v}^{(0)})]$ is denoted by $\text{CD}_k^*(\boldsymbol{\theta}, \mathbf{v}^{(0)})$. Further, we denote the average of $\text{CD}_k(\boldsymbol{\theta}, \mathbf{v}^{(0)})$ over a training set by $\overline{\text{CD}}_k(\boldsymbol{\theta})$ and its expectation by $\overline{\text{CD}}_k^*(\boldsymbol{\theta})$. The expectations are considered for theoretical reasons. They lead to deterministic updates, but are computable only for small models.

Training RBMs using CD need not lead to a maximum likelihood estimate of the model parameters. Examples of energy functions and Markov chains for which CD_1 learning does not converge are given in [13]. Yuille [14] specifies conditions under which CD learning is guaranteed to converge to the maximum likelihood solution, which need not hold for RBM training in general. Experiments comparing the quality of small RBMs trained based on CD_k^* and true likelihood maximization are presented in [6] and [7].

Refined learning algorithms. More recently, refined algorithms also based on approximating the log-likelihood via Gibbs sampling have been proposed [23]. In *Persistent Contrastive Divergence* (PCD, [2]) the sample $\mathbf{v}^{(k)}$ in the CD approximation (2) is sampled from a Markov chain defined by the RBM parameters that is independent of $\mathbf{v}^{(0)}$. This corresponds to standard CD learning without reinitializing the visible units of the Markov chain with the current training sample. It is assumed that the chain stays close to the stationary distribution if the learning rate is sufficiently small and thus the model changes only slightly between parameter updates [215]. The PCD algorithm was further refined leading to a variant called *Fast Persistent Contrastive Divergence* (FPCD, [3]). A set of additional parameters is introduced, which are only used for Gibbs sampling. The new parameters are referred to as *fast* parameters and should lead to higher mixing rates. When calculating the conditional distributions for Gibbs sampling, the regular parameters are replaced by the sum of the regular and the fast parameters. The update rule for the fast parameters is equal to that of the regular parameters, but with an independent, large learning rate and a large weight-decay parameter. Weight decay can also be used for the regular parameters, but it was suggested that regularizing just the fast weights is sufficient [3]. For details about (F)PCD we refer to the original publications [23].

Limitations of the proposed learning algorithms. Bengio and Delalleau [7] show that CD_k is an approximation of the true log-likelihood gradient by finding an expansion of the gradient that considers the k -th sample in the Gibbs chain and showing that CD_k is equal to a truncation of this expansion. Furthermore, they prove that the residual term (i.e., the bias of CD) converges to zero as k goes to infinity, and show empirically (by comparing the log-likelihood gradient and the

expectation CD_k^* in small RBMs) that the quality of CD_k as an approximation of the log-likelihood gradient decreases as the norm of the parameters increases. Anyhow, the RBMs are still able to model the considered simple target distributions. Additionally, they find that the bias of CD_k also increases with increasing number of visible units. Fischer and Igel [8] show that CD can even lead to a steady decrease of the log-likelihood during learning. This is confirmed in [9] also for PCD and FPCD. Desjardins et al. as well as Salakhutdinov [9,16] further show that using algorithms based on tempered Markov Chain Monte Carlo techniques yields better training results than Gibbs sampling.

3 Experiments

In our experiments, we study the evolution of the log-likelihood during gradient-based training of RBMs using CD_k , PCD, or FPCD. We first briefly describe our benchmark problems and then give details of the experimental setup.

Benchmark problems. We consider two artificial benchmark problems taken from the literature [12,17]. The *Labeled Shifter Ensemble* is a 19 dimensional data set containing 768 samples. The samples are generated in the following way: The states of the first 8 visible units are set uniformly at random. The states of the following 8 units are cyclically shifted copies of the first 8. The shift can be zero, one unit to the left, or one to the right and is indicated by the last three units. The log-likelihood is $768 \log \frac{1}{768} \approx -5102.43$ if the distribution of the data set is modeled perfectly. Further, we consider a smaller variant of the *Bars-and-Stripes* problem described in [17] with 16 instead of 25 visible units. Each pattern corresponds to a square of 4×4 units (if not stated otherwise) and is generated by first randomly choosing an orientation, vertical or horizontal with equal probability, and then picking the state for all units of every row or column uniformly at random. Since each of the two completely uniform patterns can be generated in two ways, the lower bound of the log-likelihood is -102.59 .

Experimental setup. The RBMs were initialized with weights drawn uniformly from $[-0.5, 0.5]$ and zero biases. The number of hidden units was chosen to be equal to, twice, or half the number of the visible units.

The models were trained on both benchmark problems using gradient ascent on CD_k with $k \in \{1, 2, 4, 10, 100\}$, PCD, or FPCD. In the experiments presented here, we only discuss batch learning, but it was verified for CD that online learning leads to similar results. The batch update rule was augmented with optional weight-decay, that is, for CD_k we have

$$\boldsymbol{\theta}^{(g+1)} = \boldsymbol{\theta}^{(g)} + \eta \overline{\text{CD}}_k(\boldsymbol{\theta}^{(g)}) - \lambda \boldsymbol{\theta}^{(g)}. \quad (3)$$

We tested different learning rates η and values of the weight-decay parameter λ (which is set to zero if not stated otherwise). Using a momentum term did not improve the results in our experiments (not shown). For the fast parameters in

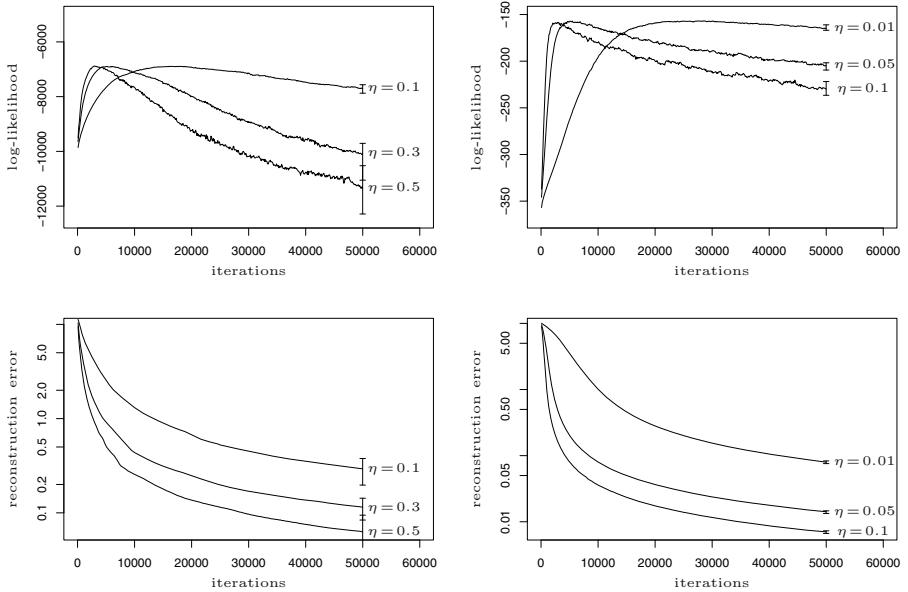


Fig. 1. Top: Evolution of log-likelihood for CD_1 using steepest-descent with different learning rates. Shown are the medians over 25 trials for the Shifter (left) and Bars-and-Stripes (right) problem, error bars indicate quartiles. Bottom: Corresponding reconstruction error.

FPCD the learning rate was set to 0.1 and the weight-decay-parameter was set to $\lambda_{\text{fast}} = \frac{19}{20}$ as suggested in [3].

In order to analyze stochastic effects of the Gibbs sampling, we also did experiments using the computationally expensive expectation $\overline{\text{CD}}_1^*(\boldsymbol{\theta}^{(g)})$ of the CD update on a further reduced Bars-and-Stripes problem with 9 visible units.

To save computation time, the exact likelihood was calculated only every 10 iterations of the learning algorithm. We additionally computed the mean reconstruction error, which has been proposed as an early stopping criterion for CD training [18][19] and is typically used to train autoassociators [18][20]. The reconstruction error for one training example \mathbf{v} is given by $-\log P(\mathbf{v}|E[\mathbf{H}|\mathbf{v}])$. All experiments were repeated 25 times and the medians of the results are presented if not stated otherwise.

Results. The evolution of the median log-likelihood for CD_1 with different learning rates is shown in Fig. II. After an initial increase, the log-likelihood steadily decreases and the model gets worse. This happens systematically in every trial as indicated by the quartiles. The higher the learning rate (i.e., the faster the learning) the more pronounced the divergence.

The increase of the negative log-likelihood was also observed if the expectation $\overline{\text{CD}}_1^*$ of the 1-step sample was used instead of a real sample. This shows that the observed effects are not caused by sampling noise. Because of computational

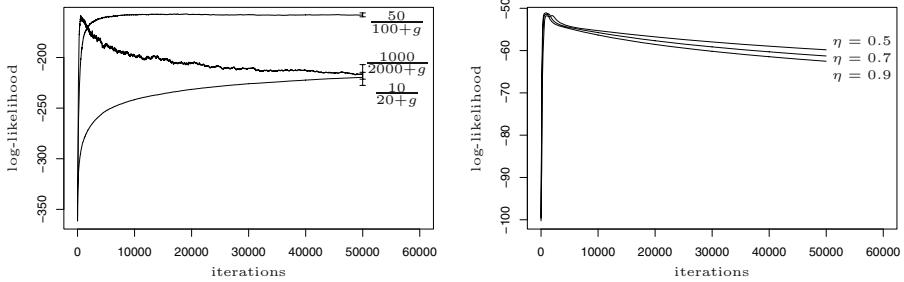


Fig. 2. Left: Log-likelihood for CD_1 with different adaptive learning rates for Bars-and-Stripes. Right: Log-likelihood for CD_1^* for the smaller Bars-and-Stripes problem with 3×3 units. In contrast to all other plots, here only single trials and not medians are shown.

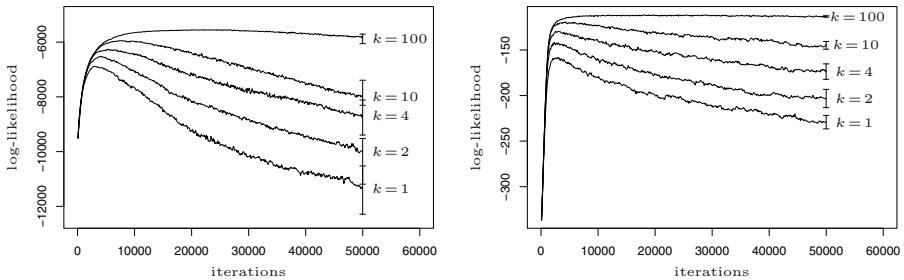


Fig. 3. Log-likelihood for CD_k with different choices of k . The learning rates were $\eta = 0.5$ and $\eta = 0.1$ for the Shifter (left) and Bars-and-Stripes (right) problem.

complexity, we performed only three single trials with different learning rates on a smaller variant of the Bars-and-Stripes problem (3×3 pixel) in the $\overline{\text{CD}}_1^*$ experiments, see right plot in Fig. 2. Without weight decay, the norm (we looked at both ∞ - and 2-norm) of the RBM parameters steadily increased (this is no surprise and therefore the results are not shown).

Comparing the course of the log-likelihood with the corresponding evolution of the reconstruction error, also shown in Fig. 1, reveals that the reconstruction error constantly decreased in our experiments. Thus, an *increase* of the reconstruction error could not be used as a criterion for early-stopping.

As shown in the left plot of Fig. 2, using a decaying learning rate $\eta^{(g)}$ (with decay schedule $\eta^{(g)} = \frac{c_1}{c_2+g}$, where $c_1, c_2 \in \mathbb{R}^+$) can prevent divergence. However, if the learning rate decays too fast (c_2 is small), learning will become too slow and if $\eta^{(g)}$ decays too slowly, the divergence is still observed.

The plots in Fig. 3 show the dependence on the number of sampling steps k . As expected, the larger k the less severe the divergence. However, in our experiments we observed a clear decrease of the likelihood even for $k = 10$. For $k = 100$, there was only a slight final decrease of the likelihood in the Shifter task and a negligible decrease in the Bars-and-Stripes benchmark.

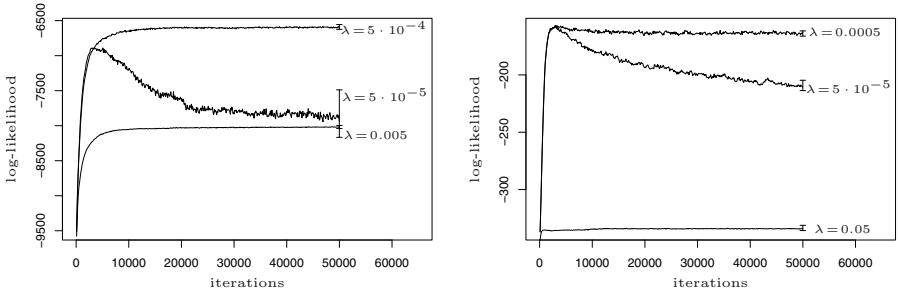


Fig. 4. Evolution of the log-likelihood for CD_1 and weight-decay with different weight-decay parameters λ and learning rates as in Fig. 3 for the Shifter (left) and Bars-and-Stripes (right) problem

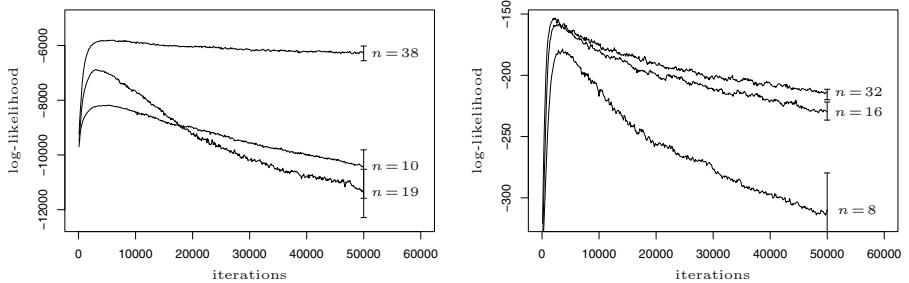


Fig. 5. Log-likelihood for CD_1 applied to RBMs with different numbers of hidden variables n for the Shifter (left) and Bars-and-Stripes (right) problem

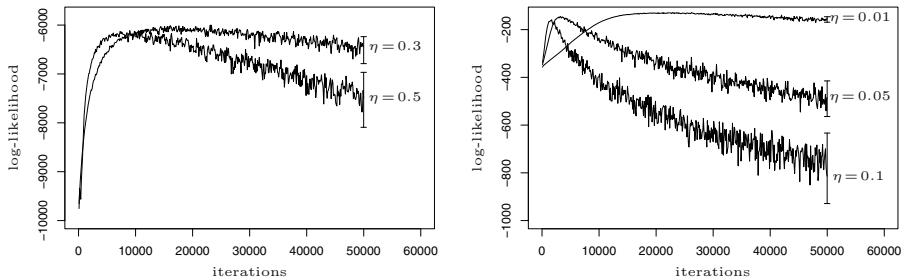


Fig. 6. Evolution of log-likelihood for PCD (Shifter, left plot) and FPCD (Bars-and-Stripes, right plot) depending on learning rate η

The effect of L_2 -norm weight-decay with different weight-decay parameters λ is shown in Fig. 4. The results indicate that the choice of the hyperparameter λ is crucial. If it was chosen correctly, the divergence problem was solved. If λ was too large, the RBMs did not model the target distribution accurately. If the weight-decay parameter was too small, it could not prevent divergence.

The influence of the number of hidden units is demonstrated in Fig. 5. The more hidden units the more expressive power the RBM has [21]. Thus, the more hidden units the “easier” the problem for the RBM. Therefore, the results in Fig. 5 suggest that the easier the problem the lower the risk of divergence.

The learning curves for PCD and FPCD were very similar to each other. We observed the same divergence effects (e.g., see Fig. 6) that could be tamed by weight-decay (weight-decay results are not shown).

4 Discussion

We have shown on benchmark problems from the literature that vanilla gradient-based optimization of RBMs via k -step CD, PCD, and FPCD can lead to a systematic decrease of the likelihood after an initial increase. The reason for this is that an increase of the model parameters increases the difference between the CD update and a gradient step on the log-likelihood. The weight increase steadily slows down the mixing rate of the Gibbs chain associated with the CD approximation (the fact that Gibbs chains in general converge faster if the start distribution is close to the target distribution does not compensate for this). With increasing weights the mixing rate goes down and makes sampling in the Gibbs chain more and more deterministic, thus inducing a strong bias. If (for some components) this bias further increases the respective weights and decreases the likelihood, CD learning diverges. However, is this really a problem in practice? One may argue that there are several well-known and simple ways to address this problem, namely (i) early stopping of the learning process [18,19], (ii) regularization using weight-decay [15,4], and (iii) increasing k dynamically when the weights increase [7].

Early stopping requires some reliable indicator that tells us when to stop and that can be computed efficiently. However, monitoring the true likelihood periodically is only possible for small problems, and the reconstruction error as discussed by [18] and [19] may gradually decrease in spite of decreasing likelihood as, for example, shown in our experiments. An adaptive learning rate can have a similar effect as early-stopping. A decaying learning rate $\eta^{(g)}$ with an appropriate schedule can prevent divergence. However, choosing the right schedule is crucial and difficult. If the learning rate decays too fast, learning will become too slow and we will not reach sufficiently good solutions in time. If $\eta^{(g)}$ decays too slowly, the learning rate adaptation has little effect and divergence is still observed. Weight decay offers a solution – if the regularization parameter is chosen correctly. If chosen too large, the model may not represent the target density accurately enough. If chosen too small, the decay term does not prevent divergence. As suggested in [7], increasing k can be a good strategy to find models with higher likelihood and it can also prevent divergence. However, divergence occurs even for values of k too large to be computationally tractable for large models. Thus, a dynamic schedule that enlarges k as the weights increase is needed [7]. Finding such a schedule is an open problem. It seems that the more difficult the problem (i.e., the more difficult it is for the RBM to represent the

target density) the more pronounced the divergence effect. The low-dimensional problems investigated in [7] are all rather easy to learn for the considered RBMs and therefore the divergence is not apparent in that study. The dependence on difficulty makes the observed phenomenon relevant for DBNs. In these multi-layer architectures, simple models such as RBMs are used in each layer and the complexity of the target distribution is reached by stacking these simple models. The lower layer(s) cannot (or should not) represent the target density alone – and thus RBMs in DBNs face distributions that are difficult to learn.

5 Conclusion

Optimization based on k -step Contrastive Divergence (CD) or (Fast) Persistent CD is a promising approach to train undirected graphical models. It has proven to be successful in challenging applications and has contributed significantly to the current revival of Restricted Boltzmann Machines (RBMs) and deep architectures. The CD is a biased estimate of the desired update direction. This bias is reduced with increasing k and gets worse with increasing norm of the model parameters (i.e., slower mixing rates of the involved Markov chain). While it is common knowledge that CD learning may only approximate the maximum likelihood solution, we showed that the bias can lead to divergence of the learning process in the sense that the model systematically and drastically gets worse if k is not large. Thus, for training algorithms relying on Gibbs sampling based stochastic approximations of the log-likelihood gradient, there is a need for robust mechanisms that control the weight growth in CD and related learning algorithms, for example, reliable heuristics for choosing the weight decay parameters or suitable criteria for early-stopping. New learning methods for RBMs using Markov Chain Monte Carlo algorithms based on tempered transitions are promising [9,16], but their learning and scaling behavior needs to be further explored.

Acknowledgements

We acknowledge support from the German Federal Ministry of Education and Research within the National Network Computational Neuroscience under grant number 01GQ0951.

References

1. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. *Neural Computation* 14, 1771–1800 (2002)
2. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) International Conference on Machine learning (ICML), pp. 1064–1071. ACM, New York (2008)
3. Tieleman, T., Hinton, G.E.: Using fast weights to improve persistent contrastive divergence. In: Pohoreckyj Danyluk, A., Bottou, L., Littman, M.L. (eds.) International Conference on Machine Learning (ICML), pp. 1033–1040. ACM, New York (2009)

4. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. *Neural Computation* 18(7), 1527–1554 (2006)
5. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* 313(5786), 504–507 (2006)
6. Carreira-Perpiñán, M.Á., Hinton, G.E.: On contrastive divergence learning. In: 10th International Workshop on Artificial Intelligence and Statistics (AISTATS 2005), pp. 59–66 (2005)
7. Bengio, Y., Delalleau, O.: Justifying and generalizing contrastive divergence. *Neural Computation* 21(6), 1601–1621 (2009)
8. Fischer, A., Igel, C.: Contrastive divergence learning may diverge when training restricted Boltzmann machines. In: Frontiers in Computational Neuroscience. Bernstein Conference on Computational Neuroscience, BCCN 2009 (2009)
9. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Dellaleau, O.: Parallel tempering for training of restricted Boltzmann machines. iN: Journal of Machine Learning Research Workshop and Conference Proceedings (AISTATS 2010), vol. 9, pp. 145–152 (2010)
10. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. In: Rumelhart, D.E., McClelland, J.L. (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations*, vol. 1, pp. 194–281. MIT Press, Cambridge (1986)
11. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. *Cognitive Science* 9, 147–169 (1985)
12. Hinton, G.E., Sejnowski, T.J.: Learning and relearning in Boltzmann machines. In: Rumelhart, D.E., McClelland, J.L. (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations*, vol. 1, pp. 282–317. MIT Press, Cambridge (1986)
13. MacKay, D.J.C.: Failures of the one-step learning algorithm. Cavendish Laboratory, Madingley Road, Cambridge CB3 0HE, UK (2001),
<http://www.cs.toronto.edu/~mackay/gbm.pdf>
14. Yuille, A.: The convergence of contrastive divergence. In: Saul, L., Weiss, Y., Bottou, L. (eds.) *Advances in Neural Processing Systems (NIPS 17)*, pp. 1593–1600 (2004)
15. Younes, L.: Maximum likelihood estimation of gibbs fields. In: Possolo, A. (ed.) *Proceedings of an AMS-IMS-SIAM Joint Conference on Spacial Statistics and Imaging. Lecture Notes Monograph Series*, Institute of Mathematical Statistics, Hayward (1991)
16. Salakhutdinov, R.: Learning in markov random fields using tempered transitions. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) *Advances in Neural Information Processing Systems*, vol. 22, pp. 1598–1606 (2009)
17. MacKay, D.J.C.: *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, Cambridge (2002)
18. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., Montreal, U.: Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) *Advances in Neural Information Processing (NIPS 19)*, pp. 153–160. MIT Press, Cambridge (2007)
19. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) *Advances in Neural Information Processing Systems (NIPS 19)*, pp. 1345–1352. MIT Press, Cambridge (2007)
20. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* 323(6088), 533–536 (1986)
21. Le Roux, N., Bengio, Y.: Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation* 20(6), 1631–1649 (2008)

Multitask Semi-supervised Learning with Constraints and Constraint Exceptions

Marco Maggini and Tiziano Papini

Dipartimento di Ingegneria dell'Informazione
Università di Siena
Via Roma 56, 53100 Siena, Italy
`{maggini,papinit}@di.unisi.it`

Abstract. Many applications require to jointly learn a set of related functions for which some a-priori mutual constraints are known. In particular, we consider a multitask learning problem in which a set of constraints among the different tasks are known to hold in *most cases*. Basically, beside a set of supervised examples provided to learn each task, we assume that some background knowledge is available in the form of functions that define the admissible configurations of the task function outputs for *almost* each input. We exploit a semi-supervised approach in which a potentially large set of unlabeled examples is used to enforce the constraints on a large region of the input space by means of a proper penalty function. However, since the constraints are known to be subject to exceptions and the inputs corresponding to these exceptions are not known a-priori, we propose to embed a selection criterion in the penalty function that reduces the constraint effect on those points that are likely to yield an exception. We report some experiments on multi-view object recognition showing the benefits of the proposed selection mechanism with respect to an uniform enforcement of the constraints.

Keywords: Semi-supervised learning, Learning with constraints, Multi-view object recognition, Neural Networks.

1 Introduction

In some learning tasks the hypothesis space can be significantly reduced by exploiting known properties of the final solution, available as background knowledge. The known properties can be directly embedded into the learning machine (see e.g. [1]) or enforced as constraints during training (see e.g. [2]). Moreover, when jointly learning multiple tasks, the training process can benefit by a multitask approach that tries to exploit the correlations among the single tasks [3].

Following the approach presented in [4] for multi-view object recognition in the case of SVM classifiers, we propose a multitask semi-supervised learning framework in which the known constraints among the tasks are enforced by adding an additional penalty term to the cost function used to define the learning goal for neural networks. The resulting learning procedure is just a straightforward extension of BackPropagation that takes into account the derivative of

the additional term in the backward step. We also discuss the problem of the presence of exceptions for the provided task constraints, that may hinder the training process. If we assume that it is not known a priori for which regions in the input space the exceptions arise, we should devise some heuristics to limit the impact of uncorrect information in the learning process. We propose the use of a *gating* function that weighs the contribution of the constraints on the basis of the outputs yielded by the current hypothesis. The idea we investigate is that it is preferable to enforce the constraints only when the network has enough confidence on its decisions.

We evaluate the proposed approach on a multi-view object recognition task as in [4] showing some promising results when using the constraints with an appropriate gating function.

The paper is organized as follows. The next section describes how the constraints among the different tasks can be embedded in the learning procedure as penalty functions. Section 3 describes the constraints that can be exploited in a multi-view object recognition system, proposing some different solutions for the gating functions. In Section 4 we present a preliminary evaluation on a multi-view recognition task on the COIL dataset and in section 5 the conclusions are drawn and the directions for future research are sketched.

2 Learning with Constraints and Constraint Exceptions

We consider a multitask learning problem in which, given an input object $X \in \mathcal{X}$, a set of related functions $\{\tau_k(X)|k = 1, \dots, T\}$ are to be learned from examples. In general, we can assume that each function $\tau_k(X)$ processes an *ad hoc* set of features extracted from the input object X , that are relevant for solving the k -th task. Thus, more precisely we can express the set of functions as $\{\tau_k(\mathbf{x}_k)|\mathbf{x}_k = \mathcal{F}_k(X), k = 1, \dots, T\}$, where $\mathcal{F}_k(X)$ is the feature extraction function designed for the k -th task and $\mathbf{x}_k \in \mathbb{R}^{n_k}$ is the k -th feature vector of dimension n_k . Since all the feature vectors are assumed to be extracted from the same input object X , they are mutually dependent, that is given a set of them the remaining ones cannot assume any configuration but their values are constrained in a given subspace. We assume that each task function computes a real value, that is $\tau_k : \mathbb{R}^{n_k} \rightarrow \mathbb{R}, k = 1, \dots, T$. This allows us to define both regression and classification tasks that can be jointly faced in the given multitask problem.

In general, the set of task functions computed on the input object X are correlated to each other. As a consequence the set of values $\{\tau_k(\mathbf{x}_k)\}$ are mutually constrained. We consider the case in which we can provide an a priori model of the existing dependences as a set of constraints on the values of the task functions $\{\tau_k(\mathbf{x}_k)\}$ expressed by a set of functions $\phi_h : \mathbb{R}^T \rightarrow \mathbb{R}$ such that

$$\phi_h(\tau_1(\mathbf{x}_1), \dots, \tau_T(\mathbf{x}_T)) \geq 0 \quad \forall X \in \mathcal{X} : \mathbf{x}_k = \mathcal{F}_k(X) \text{ and } h = 1, \dots, H. \quad (1)$$

The constraint is expressed on the functions $\tau_k(\mathbf{x}_k)$ and, hence, it is required to specify that it must hold for any valid input for these functions. It should be observed that this requirement is not for any point in the input space, i.e. the

whole space \mathbb{R}^{n_k} for the k -th function, but only for those values that represent valid features for the given problem.

The learning procedure aims at approximating each task function $\tau_k(\mathbf{x}_k)$ in an appropriate hypothesis space. In particular, we consider the case in which the functions $f_k : \mathbb{R}^{n_k} \rightarrow \mathbb{R}$, $k = 1, \dots, T$ are implemented by Multi-Layer Perceptrons but the approach can be easily extended to other learning machines. Moreover, when considering MLPs the functions, f_k can be computed either by the same network if the same input features are exploited for all the tasks (like suggested in the multitask learning literature, see e.g. [3]) or by different networks, in general. We assume a semi-supervised learning scheme in which both labeled and unlabeled examples are available. This approach has been shown to be successful when a large quantity of unlabeled examples is available and the number of supervised examples is small due to the high costs of the labeling task. Basically, the learning set will contain a set of labeled examples for each task k

$$\mathcal{L}_k = \{(\mathbf{x}_k^i, y_k^i) | i = 1, \dots, l_k\} \quad (2)$$

and a set of unsupervised examples that provide information on the feature distributions exploitable in the learning process

$$\mathcal{U} = \{(\mathbf{x}_1^i, \dots, \mathbf{x}_T^i) | i = 1, \dots, u\} \quad (3)$$

Given these two sets, the training algorithm consists in the joint optimization of the MPL weights given the following cost function

$$\begin{aligned} E[\theta_1, \dots, \theta_T] = & \sum_{k=1}^T \lambda_k^\tau \cdot \frac{1}{|\mathcal{L}_k|} \sum_{(\mathbf{x}_k^j, y_k^j) \in \mathcal{L}_k} L_k^e(f_k(\mathbf{x}_k^j | \theta_k), y_k^j) + \\ & + \sum_{h=1}^H \lambda_h^v \cdot \frac{1}{|\mathcal{U}|} \sum_{(\mathbf{x}_1^j, \dots, \mathbf{x}_T^j) \in \mathcal{U}} L_h^c(\phi_h(f_1(\mathbf{x}_1^j | \theta_1), \dots, f_T(\mathbf{x}_T^j | \theta_T))) , \end{aligned} \quad (4)$$

where $[\theta_1, \dots, \theta_T]$ collects the weights of the MLPs used to approximate the multitask set of functions \square , $L_k^e(f_k(\mathbf{x}_k | \theta_k), y_k)$ is a loss function that measures the fitting quality of $f_k(\mathbf{x}_k)$ with respect to the target y_k (a common choice for MLPs is the quadratic loss $|f_k(\mathbf{x}_k | \theta_k) - y_k|^2$), $\lambda_k^\tau > 0$ is the weight of the error for the k -th task \square , the constraint penalty loss function $L_h^c(\phi)$ is positive when the constraint is violated and zero otherwise (in the following we will use quadratic penalties), and $\lambda_h^v > 0$ is a parameter that allows us to weigh the contribution of each constraint. The definition of the cost function follows the classical penalty approach for constrained optimization, by adding a term that penalizes the constraint violation on the input space. Even if the original formulation of eq. \square

¹ We assume the more general setting where each task exploits an independent MLP, such that θ_k is the set of weights of the k -th neural network.

² If all the tasks have the same importance we can assume that this value is independent on k .

assumes hard constraints on the whole feature space, the penalty implementation relaxes this requirement in two directions. First the constraints are enforced only on a discrete sampling of the input space, which implicitly expresses the joint relationships among the considered feature representations \mathbf{x}_k of the input object X (we can assume that the unsupervised input tuples are drawn from an unknown joint distribution $p_{\mathcal{X}}(\mathbf{x}_1, \dots, \mathbf{x}_T)$ that models the correlations among the different *views* of the same object). Second the constraints are applied only by minimizing a penalty term and, given the available examples and the structure of the hypothesis space (i.e. the MLP architectures), it is not guaranteed that the constraints are exactly verified even on the available examples.

The training of the MLPs must be performed jointly since the penalty term considers the complete set of outputs. However, the classic BackPropagation algorithm can be applied to train each MLP with just a straightforward modification due to the presence of the penalty term in the cost function. In fact, this term exploits all the outputs and yields a contribution in the gradient computation that takes into account the interactions among the different functions. If we denote by $V(f_1(\mathbf{x}_1^j|\theta_1), \dots, f_T(\mathbf{x}_T^j|\theta_T))$ the penalty term for the unlabeled tuple $(\mathbf{x}_1^j, \dots, \mathbf{x}_T^j) \in \mathcal{U}$, the generalized error $\delta_k^V(j)$ to be back-propagated from the k -th output is computed by

$$\delta_k^V(j) = \frac{\partial V}{\partial f_k(\mathbf{x}_k^j)} \cdot y'_k(a_k(\mathbf{x}_k^j))$$

where $y'_k(\cdot)$ is the derivative of the neuron's output function and $a_k(\mathbf{x}_k^j)$ is the activation of the k -th output neuron. Once this value is computed for the output neuron, the further back-propagation to the hidden layer(s) and the gradient computation follows the classic BP schema.

However, sometimes the constraints can be subject to *exceptions*. This means that while we are assuming that they should be enforced for any tuple in input, it may happen that our constraint modeling is inaccurate and there exist some regions in the feature space in which the constraint does not hold. A correct modeling of the constraint would require to specify these regions but this may often be impractical since it would require a deep analysis of low level features and of their distribution among the exception and non-exception cases. Moreover, in many cases it is simpler to provide a general and clear rule even if we know that it may have exceptions. As it will be outlined in the next section, in a multi-view object recognition system it is straightforward to assume that the output for each view classifier should be coherent with the others, but we should consider that some pairs of objects may only be distinguishable from some views and not from others (i.e. the different view classifiers can provide incoherent outputs in some cases that may not be known a priori). If we know a priori the regions where the given constraints are not to be enforced, we only have to avoid to consider tuples from those regions in the definition of the penalty term. Instead, if these regions are unknown, the learning algorithm should apply a different weight to each term in the penalty by applying a criterion that should

estimate the risk of the given sample to be an exception for the constraint. We will consider a solution in which each penalty term becomes

$$G(f_1(\mathbf{x}_1^j|\theta_1), \dots, f_T(\mathbf{x}_T^j|\theta_T)) \cdot L_h^c(\phi_h(f_1(\mathbf{x}_1^j|\theta_1), \dots, f_T(\mathbf{x}_T^j|\theta_T)))$$

where $G(f_1(\mathbf{x}_1^j|\theta_1), \dots, f_T(\mathbf{x}_T^j|\theta_T))$ is a gating function which tries to estimate if the current sample is likely to be an exception. From the expression of G it is clear that the hypothesis underlying this approach is that it is possible to detect exceptions by looking at the outputs computed by the learning model. The idea, that will be clarified in the next section for a specific case, is that contradictory information exploited in the learning process is likely to drive some of the network outputs to values denoting a higher degree of uncertainty. Thus, the gating term aims at reducing the effect of the constraint on those samples for which there is no clear decision. It should be noted that the unsupervised examples on which the constraints are enforced should have an actual impact only if the supervised examples provide enough information in the region where they are enforced, in any case.

3 Multi-view Object Recognition

A multi-view object recognition system exploits a set of images of the same object acquired from different point of views. The goal is to recognize a specific object from a collection A of known items, possibly rejecting inputs not corresponding to any object of the collection. Hence, the problem can be cast as a multiclass classification task with each class corresponding to an object of the collection. The availability of multiple views allows us to employ a richer input description that, for instance, can be used to design a committee of classifiers each processing one view.

If the T views are acquired by different cameras (for instance, spaced at equal angular distances around the object), each object will be described by a *viewset* $X = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$, such that \mathbf{x}_i is the set of features extracted from the image acquired by the i -th camera [4]. Clearly, the feature extraction process $\mathcal{F}_i(X)$, that provides the i -th object representation as viewed by the i -th camera, depends on many factors such as the camera characteristics, the mutual position between the camera and the object, the environment properties (e.g. light conditions) and the type of features we decided to use (gray levels, transforms, invariant descriptors, etc.).

We approach the multiclass problem, by adopting a modular architecture in which $|A|$ binary multi-view classifiers are trained, taking the final decision with a one-against-all strategy with threshold for rejection. The rejection threshold thr_i for each object i is determined by optimizing the accuracy on a validation set, and it is used to scale the classifier output in order to perform a fair one-against-all comparison. In particular, if $c_i(X)$ is the output of the i -th classifier, we compute its normalized value as $c_i^n(X) = (c_i(X) - thr_i)/(1 - thr_i)$. All the $c_i^n(X)$ assume values in $[-1, 1]$ and have the same rejection threshold equal

to 0. Hence, if $\max_i c_i^n(X) \leq 0$ the input X is rejected, otherwise the object corresponding to the maximum $c_i^n(X)$ is recognized.

There are many different ways to exploit the available views in the design of the classifiers. A simple approach is to concatenate all the feature vectors into an unique vector that is exploited as input. This approach can increase the complexity of the required solution since the input dimensionality may grow substantially. A simpler approach is to design a committee of classifiers, each trained to recognize the object from a specific point of view. In this case a proper combination function should be applied to combine the outputs of the view classifiers into a single value. Even if more sophisticated techniques could be adopted, we used a simple average of the view classifier outputs

$$c_i(X) = \frac{1}{T} \sum_{k=1}^T f_{i,k}(x_k) . \quad (5)$$

In particular, we considered the case in which each view classifier $f_{i,k}(x_k)$ is implemented by an MLP having one sigmoidal output.

When training the set of view classifiers for a given object i , we can enforce the output coherence constraint that states that all the views should yield the same output value. In details, it should hold

$$f_{i,1}(x_1) = f_{i,2}(x_2) = \dots = f_{i,T}(x_T) .$$

This constraint is intuitive and simple but it may be subject to many exceptions since, if we consider two objects that look similar from a given point of view, the outputs for that view may receive contradictory information. In order to simplify the constraint expression (the complete set of constraints would require a comparison between any pair of view functions), we consider only “adjacent” views by adopting the following penalty term

$$\sum_{k=1}^T |f_{i,k\%T+1}(x_{k\%T+1}) - f_{i,k}(x_k)|^2 \quad (6)$$

where we denoted the modulo operator by $\%$. Since we are dealing with a set of binary classification problems, we tried also to define “logic” coherence constraints between pairs of outputs. To express this constraint we used the extension of logic operators to real valued variables defined in the interval $[0, 1]$ as suggested by the definition of T-norms used in Fuzzy Logic [5]. In particular, we can exploit the product T-norm which models the logic AND of two variables by their product $T(a, b) = ab$, and the OR by a generalization of the De Morgan’s law, computed as $1 - (1 - a)(1 - b)$. Hence the equality of two logic variables, that is expressed by the logic rule $(a \wedge b) \vee (\neg a \wedge \neg b)$, can be implemented as $1 - [1 - ab][1 - (1 - a)(1 - b)]$ when considering real valued truth degrees in $[0, 1]$. In this case, since each logic constraint must be true, i.e. equal to 1, we can enforce the penalty

$$\sum_{k=1}^T [1 - f_{i,k\%T+1}(x_{k\%T+1}) \cdot f_{i,k}(x_k)] \cdot [1 - (1 - f_{i,k\%T+1}(x_{k\%T+1})) \cdot (1 - f_{i,k}(x_k))] . \quad (7)$$

Finally, in order to implement the gate function $G(\cdot)$ used to weigh the constraints in case of exceptions, we first define a function that classifies the network output as uncertain or not. This function is defined as

$$g(x) = 1 - e^{-(\frac{x-0.5}{0.2})^2} \quad (8)$$

and is plotted in Figure 1.

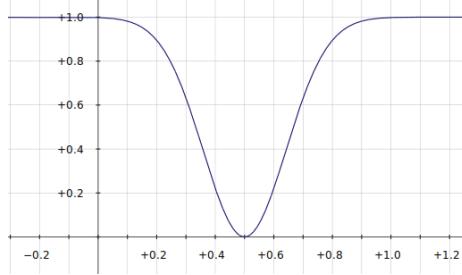


Fig. 1. Function for testing the certainty of the network output

Basically, the output is considered not certain when $x = 0.5$ and certain when it assumes values close to 1 or 0. Using this definition, we can implement different gating functions based on different ways of combining the certainty of the outputs considered in the constraints. For instance, we can decide to apply the constraint only when all the variables are certain (i.e. they are combined by a logic AND) or when at least one is certain (i.e. we exploit the logic OR). In particular, if we use the product T-norm, we can define the following “gated” versions of the original cost of eq. 6: the certainty AND penalty function is

$$\sum_{k=1}^T g(f_{i,k\%T+1}(x_{k\%T+1})) \cdot g(f_{i,k}(x_k)) \cdot |f_{i,k\%T+1}(x_{k\%T+1}) - f_{i,k}(x_k)|^2 \quad (9)$$

while the certainty OR penalty cost is

$$\sum_{k=1}^T [1 - (1 - g(f_{i,k\%T+1}(x_{k\%T+1}))) \cdot (1 - g(f_{i,k}(x_k)))] \cdot |f_{i,k\%T+1}(x_{k\%T+1}) - f_{i,k}(x_k)|^2 . \quad (10)$$

Clearly, it is possible to define other gating criteria, but these are the ones we evaluated in the experiments reported in the next section.

4 Experimental Results

We exploited the COIL100 (Columbia Object Image Library) dataset to evaluate the proposed multitask learning procedure for multi-view object recognition. The original dataset consists of color images of 100 3D objects that were obtained by placing the object on a turntable and by acquiring an image every 5° of rotation. Hence, a total of 72 views at the resolution of 128x128 pixels are available for each object (7200 images in total). Figure 2 shows some examples of the objects contained in the dataset. This dataset has been extensively used to compare object recognition algorithms [6,7,8,9,10,11,12,13] but it has been shown that the object recognition task on this dataset can yield high accuracy just using the average color [14,15]. Hence, in order to increase the difficulty of the task, we applied the same preprocessing proposed in [4] by rescaling each image to the size 32x32 pixels and considering only gray levels instead of colors.

In the considered experimental setting, multi-view recognition is performed by processing a set of 4 images captured at 90° of rotation from each other. We assume that the images can be acquired with a potential point-of-view misalignment in the range $[-45^\circ, 45^\circ]$. Among the available objects, we decided to split the into a known object collection K , containing the first 20 objects, and an unknown object set U , containing the remaining 80 objects. The classifier is composed by 20 neural networks each trained to recognize a given object in K as described in the previous section. The set U is used to test the rejection performance of the trained classifier. The characteristics of the sets used for learning and testing are summarized in Table 1. Given the small number of examples used for training, the constraints were enforced also on validation data.

The penalty weight λ and the number of hidden neurons were chosen to optimize the accuracy on the validation set. λ was varied in $\{0.2, 0.4, 0.6, 0.8, 1, 2\}$



Fig. 2. Sample images from the COIL-100 database

Table 1. Partition of the 7200 images of the COIL-100 into training, validation and test sets. The left portion of the table details the list of objects and total number of images in the considered set, whereas the right one collects information on the viewsets. With “the remaining ones” we refer to the viewsets not included in training and validation sets.

Set	Objects	Images
<i>Training</i>	$1, \dots, 20$	400
<i>Validation</i>	$1, \dots, 20$	240
<i>Test K</i>	$1, \dots, 20$	800
<i>Test KU</i>	$1, \dots, 20$ $21, \dots, 100$	800 1440

Set	Viewsets	Positions
<i>Training</i>	5	$\{-30, -5, 0, +5, +30\}$
<i>Validation</i>	3	$\{-15, +15, +45\}$
<i>Test K</i>	10	The remaining ones
<i>Test KU</i>	10 18	The remaining ones All

Table 2. Recognition (macro) accuracies when training with different constraints

ANN	Train	Validation	Test K	Test KU
without constraints	100%	96.67%	99.5%	73.9%
with constraints	100%	98.33%	100%	73.48%
with constraints (T-norm)	100%	98.33%	100%	75.06%
with constraints (AND)	100%	98.33%	100%	79.39%
with constraints (OR)	99%	58.33%	84.50%	88.35%

Table 3. Recognition (macro) accuracies of the proposed approaches in the test set (K) when one of the views is not active

Enable View	Test K				Test KU			
	1	2	3	4	without constraints	with constraints (AND)	without constraints	with constraints (AND)
no yes yes yes	97.5%				98%		74.27%	75%
yes no yes yes	98%				99.5%		79.45%	78.72%
yes yes no yes	98%				98.5%		71.77%	75.79%
yes yes yes no	98%				100%		69.39%	80.61%

and the number of hidden neurons in $\{5, 10, 15, 20, 25, 30, 35, 40\}$. The neural network was fed directly with the 1024 gray levels (normalized to be in $[-1, 1]$) of the 32x32 image.

Table 2 reports the micro accuracies obtained when training the neural networks without constraints, exploiting the quadratic constraint of eq. 6, the product T-norm constraint of eq. 7, the quadratic constraint with the AND gating function of eq. 9, and the quadratic constraint with the OR gating function of eq. 10. Most of the tested classifiers yield an almost perfect recognition rate on the test K, with a slight improvement when using the constraints. However, the effect of the constraints is more evident when considering the test KU, for which the networks trained with constraints and the gating functions are more robust with respect to the misclassification of unknown objects. The worst performances of the OR gating function show that the choice of an appropriate penalty function is crucial.

In the following experiments, we analyzed only the best performing scheme (quadratic constraint with the AND gating). First we tested the effect of each view on the recognition accuracy. As shown in table 3, some views are more important than others, especially when facing the rejection test.

Then, we tested the trained network removing the assumption that the orientation of the object is known. In this case, we need to feed the classifier with 4 different inputs obtained by “rotating” the feature vectors corresponding to the 4 input views. This corresponds to consider beside the original position, the configurations equivalent to a rotation of 90° , 180° , and 270° . In this case the decision is taken on the basis of the maximum of the four output values obtained for each input configuration. In this case the performances are lower on the rejection test: the network trained without constraints yields 98.5% accuracy on Test K and 60.5% on Test KU; the network trained with constraints and the AND gating obtains 100% accuracy on Test K and 68.5% on Test KU.

5 Conclusions and Future Work

In this paper we proposed a technique to incorporate constraints with potential exceptions into multitask semi-supervised learning with neural networks. The constraints are enforced by adding a penalty term to the cost function and, consequently, the training procedure is just a straightforward modification of the classic BackPropagation algorithm. A gating function weighting the contribution of the constraint penalty allows us to deal with the presence of exceptions. We reported a preliminary evaluation on a multi-view object recognition task showing some benefits when using the proposed technique. Future work will be devoted to a more accurate analysis of the gating functions and to the extension of the constraints to more general classes of rules.

References

1. Shawe-Taylor, J.: Symmetries and discriminability in feedforward network architectures. *IEEE Transactions on Neural Networks* 4(5), 816–826 (1993)
2. Abu-Mostafa, Y.S.: Hints. *Neural Computation* 7(4), 639–671 (1995)
3. Caruana, R.: Multitask learning. *Machine Learning* 28(1), 41–75 (1997)
4. 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. LNCS, vol. 5769, pp. 653–662. Springer, Heidelberg (2009)
5. Klir, G., Yuan, B.: Fuzzy sets and fuzzy logic: theory and applications. Prentice-Hall, Upper Saddle River (1995)
6. Murase, H., Nayar, S.: Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision* 14(1), 5–24 (1995)
7. Pontil, M., Verri, A.: Support vector machines for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(6), 637–646 (1998)
8. Mori, G., Belongie, S., Malik, J.: Shape Contexts Enable Efficient Retrieval of Similar Shapes. In: Proc. of Int. Conf. on CVPR, vol. 1, pp. 723–730 (2001)
9. Liu, X., Srivastava, A.: 3D object recognition using perceptual components. In: Proc. of Int. Joint Conf. on Neural Networks, vol. 1, pp. 553–558 (2001)
10. Obdrzalek, S., Matas, J.: Object recognition using local affine frames on distinguished regions. In: Proc. of British Machine Vision, vol. 1, pp. 113–122 (2002)
11. Obdrzalek, S., Matas, J.: Sub-linear indexing for large scale object recognition. In: Proc. of the British Machine Vision Conf., vol. 1, pp. 1–10 (2005)
12. Lyu, S.: Mercer Kernels for Object Recognition with Local Features. In: Proc. of Int. Conf. on CVPR, vol. 2, pp. 223–229 (2005)
13. Christoudias, C.M., Urtasun, R., Darrell, T.: Unsupervised feature selection via distributed coding for multi-view object recognition. In: Proc. of CVPR, pp. 1–8 (2008)
14. Roobaert, D., Van Hulle, M.: View-based 3D object recognition with support vector machines. In: Neural Networks for Signal Processing, pp. 77–84 (1999)
15. Caputo, B., Dorko, G.: How to Combine Color and Shape Information for 3D Object Recognition: Kernels do the Trick. In: Advances in NIPS, pp. 1399–1406 (2003)

Tumble Tree – Reducing Complexity of the Growing Cells Approach

Hendrik Annuth and Christian-A. Bohn

Wedel University of Applied Sciences
Wedel, FR Germany

Abstract. We propose a data structure that decreases complexity of unsupervised competitive learning algorithms which are based on the *growing cells structures* approach. The idea is based on a novel way of ordering the cells in a tree like data structure in a way that random access during training is replaced by tree traversals.

Overall time complexity is reduced from $O(n^2)$ to $O(n \log n)$ which opens new application fields to the growing cells structures approach.

Keywords: neural networks, unsupervised learning, reinforcement learning, growing cells structures, growing neural gas.

1 Introduction

The importance of automatic analysis and interpretation of complex, huge and arbitrary data sets has notably increased in the field of computer science. This is driven by the fact that these data sets nowadays can easily be created, retrieved and stored.

The analysis and interpretation of such data using unsupervised competitive learning algorithms has shown to be a great success. One important approach from the field of unsupervised learning is the growing cells structures (GCS) concept [1]. It is deduced from Kohonen's *self organizing map* (SOM) [2] and shares its advantages like autonomy, robustness, scalability and the ability of retrieving information from very complex data. In contrast the GCS approach is able to locally and globally adapt to a given problem under consideration by adding and removing cells in the network. This makes the method very flexible and adaptive. The GCS approach achieves extraordinary results in the area of classification, clustering, dimensionality reduction and data mining, such that there exist a huge amount of applications.

Despite the success of the GCS approach, due to its quadratic complexity, the restricted network size hinders many applications where huge amounts of data arise. This comes from the fact that at every iteration cycle each network cell has to be visited once and a local counter at each cell has to be decreased or increased. Up to now, approaches which try to circumvent this problem, like [3], mostly fail if input data grows significantly and complexity issues surface again.

Previous Work. A very important preparatory work for the GCS approach was [2]. They propose the self organizing map which iteratively adapts a 2D mesh to a distribution of samples. While a SOM has a fixed number of elements, the growing cells structures approach [14] allows the network for dynamically adapting to the complexity of the sample data.

Optimizations of the basic GCS approach have been presented in [5] where an advanced clustering algorithm enables analysis of huge data sets. The ability of handling high-dimensional and very noisy data sets has been proven in the field of medicine [67]. Further useful example applications of the GCS approach are document categorization [8]. And [9] describes an interesting approach supporting classification of measurements from a technical gas sensor.

Further development has been driven by the application of surface reconstruction from scanned 3D point sets. This comes from the fact that complexity issues are very significant in this area since the amount of training samples easily exceeds hundreds of thousands of samples per application case. Surface reconstruction with GCS have first been tackled in [10] and [11]. Based on this, [12] proposes a mesh optimization algorithm, and to achieve a better surface quality [13] introduces an edge swap operation. In [14] surface reconstruction is further optimized by the *smart growing cells* (SGC) network which proposes a framework of extending general unsupervised learning by application related rules.

In the following, we first explain the vital steps of the GCS algorithm, which are important to analyze runtime issues. Then we introduce the *tumble tree* data structure which reduces the complexity of the standard GCS approach. Finally, we show that this is a major breakthrough for the GCS method through its application to surface reconstruction where execution time is reduced from several weeks to just hours.

2 The Tumble Tree Idea

2.1 Standard GCS Approach

Basic Loop. The basic training process of the growing cells approach is exposed in Fig. 1.

Growing and shrinking of the network within the GCS process depends on the signal counters at each cell. Every time a cell is moved to match a sample its signal counter is affected. In [1] α is constant, but for huge data sets adaption of α during network training is required as shown in [3].

See [1] for a detailed description of the basic GCS approach.

Handling Signal Counters. If a sample is presented to the network, finding the relevant BMU (step 1 of Fig. 1) has $O(\log n)$ time complexity with n the number of cells and if the cells are structured in an octree, for example. Nevertheless, adapting the signal counters τ_k and finding the cell with the highest and those cells with a low counter for adaption of the network size (step 3 and 4 in Fig. 1) has time complexity of $O(n)$.

1. Select a random sample from the data set and find the network cell lying closest to it — the *winning* or *best matching unit* (BMU). Move the BMU and its neighbors according to a certain amount in the direction of the sample.
2. Increment the *signal counter* τ_{BMU} of the BMU and decrease signal counters of all other cells by multiplying them with a coefficient $\alpha < 1$.
3. After a certain number of iterations (steps 1 and 2) determine the cell with the greatest signal counter and add a new cell next to it, since a high signal counter indicates an underrepresentation of cells compared to the amount of samples at that cell.
4. After a certain number of iterations of step 3 determine those network cells which contain a signal counter value below a certain threshold $\overline{\tau}$ and delete them, since they are not represented through an adequate number of samples at that location.

Fig. 1. Algorithm of the classical GCS approach

Since during learning the GCS grows to about the size of the sample set, the above steps have to be executed n times leading to complexities of $O(n \log n)$ and $O(n^2)$, respectively, leading to overall complexity of $O(n^2)$. See Fig. 2 for a visualization of what happens with the signal counters if cells are moved.

This quadratic complexity is the reason for the lack of applications with sample set sizes above few hundreds of thousands of samples. From our experiences, we learned that calculation time rises from a few minutes to several weeks if network sizes rise from about 10^5 to just about 10^6 cells.

Thus, for these two tasks of updating signal counters and searching cells (steps 2, 3, and 4 from Fig. 1) we propose a new algorithm as follows.

2.2 Signal Counters in the Tumble Tree

We add a concept which we term the *tumble tree*. The tumble tree is a tree-like data structure concurrent to the GCS network with n nodes each of them linked to a single network cell — each cell in the GCS network has its counterpart in the tumble tree. Essentially, the tumble tree is a binary tree — each node k has two children. The child subtree on the left contains only nodes with a signal counter smaller than τ_k , those on the right have signal counters which are greater than or equal to that of node k .

All nodes in the tree contain a cell signal counter and a local value α_k . To determine a signal counter at a node, all α_k above the node have to be multiplied by it. The reason for that will be explained on the following pages.

Considering the algorithm from Fig. 1, our method starts again by finding the BMU (step 1), but for updating the signal counter and for searching the highest

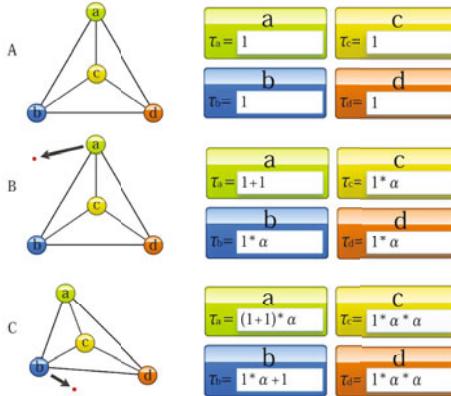


Fig. 2. On the left, a simple GCS network is exposed. In the middle, cell (a) is moved, at the bottom cell (b) is moved. The right side shows the change of the signal counters τ_k of each node. In case (A) all τ_k equal one, in (B) node (a) is moved while incrementing its signal counter, the remaining nodes' τ_k are decreased by a multiplication with α . In case (C) the similar happens with node (b).

and the low signal counters (steps 2, 3, and 4) it switches to the tumble tree and accomplishes the operation from there.

Searching Cells regarding its Signal Counters. The first of the two tasks to be realized in the tumble tree is determining the cells with the highest and the lowest signal counters. In the standard GCS method this is achieved by a linear walk through a simple list of the cells. In our proposal, we assume at the moment that the tumble tree nodes contain the correct signal counter values. Then, finding the highest τ_k can be simplified by choosing the outer right node at the bottom of the tree.

Determining all nodes with τ_k below a certain threshold $\bar{\tau}$ is accomplished by descending the tree from the root node and comparing the signal counter τ_k of the visited node k with the threshold $\bar{\tau}$. As long as $\tau_k < \bar{\tau}$ hold we take a branch to the right. If $\tau_k < \bar{\tau}$ does not hold, then the left child of the actual node is the root node of a subtree which only contains nodes of cells for which $\tau_k < \bar{\tau}$ hold. Obviously, the complexity of the traversal — the search task — is $O(\log n)$.

Topology Maintanance. If a BMU of the GCS has been determined, its signal counter has to be incremented and those of all other cells have to be decreased by a certain factor α . This could change the validity of the tumble tree since its topology is determined from the size of each τ_k .

To keep the tumble tree valid, nodes where signal counters change must be repositioned in the tree. This means the BMU's node must be taken off and be reinserted regarding its new τ_k . The remaining nodes can be kept untouched since their signal counters change to the same amount each.

Reinserting the BMU's node with a certain new τ_{BMU} is done by traversing the tree from the root node and comparing its τ_{BMU} with the signal counters at the visited nodes. Depending on the result a branch to the left or to the right child is taken until the leaves of the tree are reached. At this position the BMU's node is inserted as a new leave node.

The complexity of this step is also $O(\log n)$.

Up to now we just assumed that signal counter values are known. Finally, we explain their management in the following.

Handling Signal Counters. Up to now, the only task left with complexity $O(n)$ is the update of the signal counters in all cells after a new BMU has been determined. We solve this issue as follows.

The signal counters of all cells but the BMU must be multiplied by an actual value α . This multiplication is postponed, and instead the relevant cells are marked by multiplying this α with a node-local α_k contained in the root node of the subtree of those cells. This subtree is just the complete tumble tree without the BMU node.

Postponing this multiplication through one root node prevents from visiting all cells at the moment, but if one τ_k in a node is needed, it first has to be generated from the postponed α -values spread over the whole tumble tree. It is required for each of the following three cases.

1. If a BMU node is to be taken out of the tree, then for later reinsertion τ_{BMU} must be known. This is accomplished by determining the path back to the root and multiplying all visited (postponed) α_k with the actual τ_{BMU} . Now the BMU node with a valid signal counter can be taken out of the tree. What remains is the α -value which is contained in and which vanishes with the node. It virtually serves as postponed α -value for the subtrees and must be propagated to the left and the right child of the BMU node in order keep the signal counters underneath the missing node being valid.
2. When reinserting a BMU's node into the tumble tree again, then for the comparison, all signal counters on the way down to the right place of the node are required. This is simply accomplished by accumulating the values α_k of all nodes which are visited on the way down.

Additionally, the α -values are propagated through the nodes by multiplying them with each τ_k , and finally they are set to 1, such that nodes on this path contain the true τ_k and a neutral α_k . The reason for that is, that the node which is to be reinserted even contains a valid signal counter (from step 1 from above). Keeping the old α_k in the nodes above would virtually change the signal counter of the new node.

Additionally, since all α_k on the way down to the inserted node are combined with the relevant τ_k — and thus set to a neutral value — the former postponed α -values are now missing at the child subtrees of each of the nodes. To solve this, the propagated α_k are also propagated to the left and the right child nodes on the way down to the new node.¹

¹ The way down the tree looks like “tumbling” from the left to the right which delivers the name of our approach.

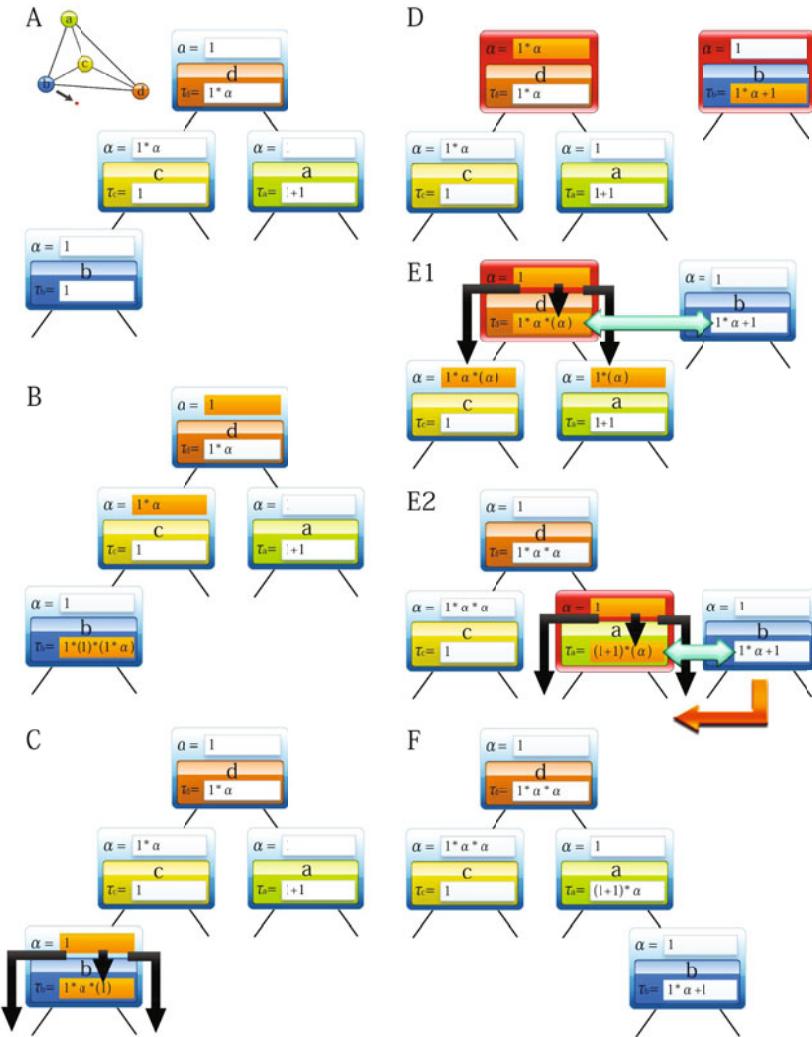


Fig. 3. A: The cell (a) has recently been selected as winning cell. Now (b) is the winner cell and the signal counter update is imminent. B: All alpha values from nodes above (b) are collected and multiplied with the signal counter of (b). C: Since (b) will be detached soon, the children of (b) must receive an update of their α -values. Therefore the update information is propagated to the neighboring nodes and α_k of (b) is set to 1. D: The signal counter of (b) is now valid and does not contain any postponed α -values. It can be detached from the network and its signal counter be incremented since it has been selected as BMU. The remaining cells are multiplied with α by postponing the multiplication in the α -value of node (d). E: The node (b) is reinserted. To determine its position its signal counter is compared with the counters of the visited cells, but before, each of the cell counters have to be determined by propagating the α_k down the tree. Then, the node will be remounted in the tree when it reaches an empty node. F: (b) has been reinserted and the signal counter update process is finished.

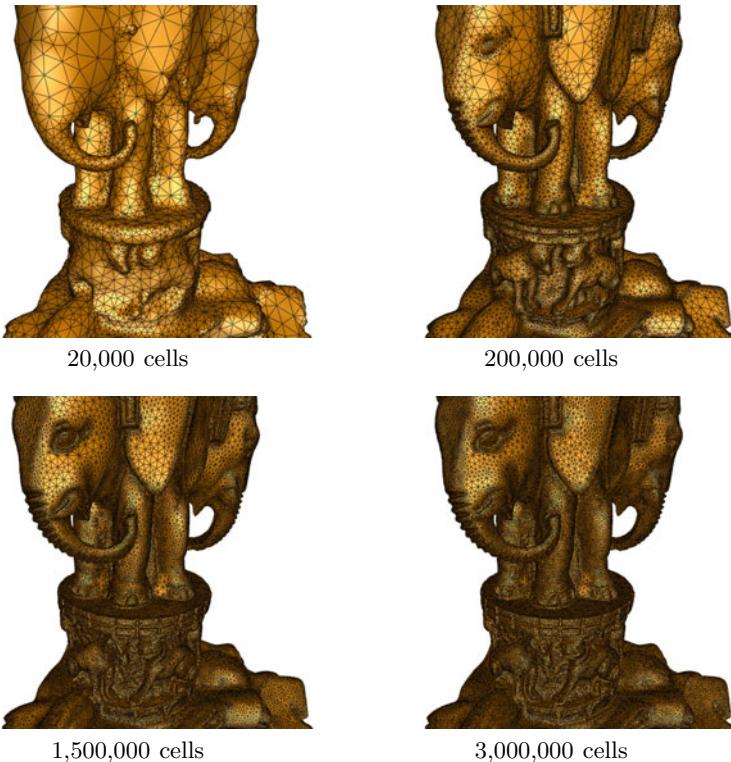


Fig. 4. A vital example application which emphasizes the importance of allowing the GCS network for growing to sizes of several millions of cells is surface reconstruction from arbitrary 3D point samples. Here, the quality of the reconstructed surface strongly depends on the number of triangles. When using a GCS for this tasks the number of cells equals the number of triangle vertices and thus the application is simply not possible being realized if few hundreds of thousands of samples (which is a typical amount in this type of tasks) require several days of computation time.

3. The highest and the low signal counters are also determined on the fly by accumulating the α_k of the visited nodes while traversing the tumble tree like described above.

All three tasks from above can be accomplished in $O(\log n)$ time complexity and without touching every node in the GCS. For an example of a simple network see Fig. 3.

Thus, the linear complexity for accessing the signal counters is now driven down to $O(\log n)$, and with complexity of step 1 in Fig. 1 this leads to an overall complexity of our GCS approach of $O(n \log n)$.

Table 1. Computing time of the classical GCS method rises quadratically with network size. Using the proposed tumble tree reduces complexity to $O(n \log n)$ and calculation times are significantly diminished in a way that an application depending on a huge number of cells can now be realized in an acceptable amount of time of few hours instead of weeks.

	Network size [number of cells]				
	20k	50k	200k	1,500k	3,000k
Classical GCS	5 min	24 min	7 h	2 weeks	8 weeks
GCS with Tumble Tree	1 min	3 min	14 min	3 h	7 h

3 Conclusion

One could think that there is nothing to add at “the algorithm is driven down from $O(n^2)$ to $O(n \log n)$ ” — nevertheless, in many cases n is not big enough to have a favorable effect on an application. Due to that, we justify our algorithm with the task of surface reconstruction which is a vital application for the GCS approach like mentioned in section 1. To assure logarithmic complexity we used a *red-black-tree* [15] as basis for the tumble tree.

Table 1 exposes the results. The acceleration of our approach compared to using the classical GCS algorithm with squared complexity is impressive. It can be seen that computation times rise dramatically if the tumble tree is not utilized. Our approach enables creating networks with several millions of cells in an acceptable time. Besides the numbers, Fig. 4 shows some visual results.

Summarizing, we showed how to reduce complexity of the common GCS approach from $O(n^2)$ to $O(n \log n)$. Essentially this is achieved by organizing the network cells in a tree-like data structure, and instead of a linear loop through the set of cells for certain operations a tree-traversal can be accomplished which requires logarithmic time complexity.

As an example, we presented an application case which was not possible being realized up to now. Since the GCS algorithm is utilized in a huge variety of applications, we are confident, that this approach might be more than simply an optimization of a classical method.

For future work, we are planning a parallelized version of the algorithm. Since even the general GCS algorithm is suitable for parallelization, we think we could achieve interactive or real-time training rates — at least for specific simplified tasks.

References

1. Fritzke, B.: Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks* 7, 1441–1460 (1993)
2. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* 43, 59–69 (1982)

3. Ivriessimtzis, I., Jeong, W.K., Lee, S., Lee, Y., Seidel, H.P.: Neural meshes: surface reconstruction with a learning algorithm. Research Report MPI-I-2004-4-005, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany (October 2004)
4. Fritzke, B.: A growing neural gas network learns topologies. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) *Advances in Neural Information Processing Systems*, vol. 7, pp. 625–632. MIT Press, Cambridge (1995)
5. Hodge, V.J., Austin, J.: Hierarchical growing cell structures: Treegcs. *IEEE Transactions on Knowledge and Data Engineering* 13, 207–218 (2001)
6. Wong, J.W.H., Cartwright, H.M.: Deterministic projection by growing cell structure networks for visualization of high-dimensionality datasets. *J. of Biomedical Informatics* 38(4), 322–330 (2005)
7. Mahony, S., Benos, P.V., Smith, T.J., Golden, A.: Self-organizing neural networks to support the discovery of dna-binding motifs. *Neural Netw.* 19(6), 950–962 (2006)
8. Deng, W., Wu, W.: Document categorization and retrieval using semantic micro-features and growing cell structures. In: *DEXA 2001: Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, Washington, DC, USA, p. 270. IEEE Computer Society, Los Alamitos(2001)
9. Cheng, G., Zell, A.: Externally growing cell structures for data evaluation of chemical gas sensors. *Neural Computing and Applications* 10(1), 89–97 (2001)
10. Hoffmann, M., Vrady, L.: Free-form surfaces for scattered data by neural networks. *Journal for Geometry and Graphics* 2, 1–6 (1998)
11. Yu, Y.: Surface reconstruction from unorganized points using self-organizing neural networks yizhou yu. In: *IEEE Visualization 1999, Conference Proceedings*, pp. 61–64 (1999)
12. Álvarez, R., Noguera, J.V., Tortosa, L., Zamora, A.: A mesh optimization algorithm based on neural networks. *Inf. Sci.* 177(23), 5347–5364 (2007)
13. Mari, Joá.F., Saito, J.H., Poli, G., Zorzan, M.R., Levada, A.L.M.: Improving the neural meshes algorithm for 3d surface reconstruction with edge swap operations. In: *SAC 2008: Proceedings of the 2008 ACM symposium on Applied computing*, pp. 1236–1240. ACM, New York (2008)
14. Annuth, H., Bohn, C.-A.: Growing cells meshing. In: *Proceedings of 3IA — The 13th International Conference on Computer Graphics and Artificial Intelligence* (2010)
15. Bayer, R.: Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta Inf.* 1, 290–306 (1972)

Sentence Extraction by Graph Neural Networks

Donatella Muratore¹, Markus Hagenbuchner²,
Franco Scarselli¹, and Ah Chung Tsoi³

¹ University of Siena, Siena, Italy

² University of Wollongong, Wollongong, Australia

³ Hong Kong Baptist University, Hong Kong

Abstract. In this paper, we will apply a recently proposed connectionist model, namely, the Graph Neural Network, for processing the graph formed by considering each sentence in a document as a node and the relationship between two sentences as an edge. Using commonly accepted evaluation protocols, the ROGUE toolkit, the technique was applied to two text summarization benchmarks, namely DUC–2001 and DUC–2002 respectively. It is found that the results obtained are comparable to the best results achieved using other techniques.

Keywords: Sentence Extraction, Text Summarization, Graph Neural Network, TextRank.

1 Introduction

With the increasing number of documents which are “born digital”, there is an urgent need to be able to summarize the documents automatically so that readers can reach relevant information of interest rapidly. Automatic document summarization can be tackled using two different strategies: extraction and abstraction. The abstraction approach is based on a complete rewriting of the input document and can produce more coherent results. This requires semantic analysis of the text, and then to summarize the semantic network into a smaller one. Thus, the abstraction process is often quite expensive computationally. On the other hand, the sentence extraction (SE) approach is a low cost solution to build summaries which are sufficiently intelligible and useful for comprehension by humans. The SE approach identifies and concatenate the most relevant units¹ in textual documents [1]. In general, in the SE approach the input document is analyzed and the information relevance of each sentence is measured; then, the sentences are sorted according to their importance and the most important ones are extracted and used in the summary. Such procedures depend on the evaluation criteria used to measure the relevance of the sentences. For example, the criteria proposed in literature are based on deep natural language processing

¹ In theory, units with different levels of granularity can be considered: keywords, sentences, paragraphs. However, most researchers concentrate their attention on sentence extraction, since the readability of a list of keywords is typically low, whereas it is difficult to build short summaries using paragraphs.

and heuristics [23] or on simple language analysis and naive Bayes [4], decision trees [5], neural networks [6], hidden Markov models [7].

Moreover, even if most of the approaches measure the importance of each sentence focusing on features extracted from the sentence itself, a well known method, called TextRank, evaluates a sentence on the basis of its similarity to other units in the same document [8]. The main idea underlining such a solution is that the most valuable text units are those that share the largest number of concepts with the other units. In this case, the document is represented by a graph, where the nodes stand for the sentences and the links indicate the similarities between the sentences. A link is quantified by the degree of similarity and it is computed by considering the common words that the sentences share. The iterative importance ranking algorithm used on the graph is similar to the PageRank algorithm², though in TextRank the link information takes part in the computation [8].

An important limitation of TextRank and its generalizations is that those algorithms are not adaptive and their performance may be dependent strongly on the document domain under consideration, the adopted measure of sentence correlation, and the parameters of PageRank that are used. Moreover, PageRank is a very specific computational paradigm and it may not able to model the complex behavior required to solve the sentence extraction task. In this paper, we present a connectionist approach that can overcome those limitations. The method is based on *Graph Neural Networks* (GNNs) [10], a recently proposed neural network architecture that is able to directly process graphs having labels, which, in this case, can contain any feature of the sentences that can be useful for the considered task. The GNN training algorithm allows to learn from examples to produce a desired output on each node, so that GNNs can automatically learn to predict the importance of each sentence. Moreover, it has been proven that GNNs satisfy a kind of universal approximation property and they can implement a very wide class of functions on graphs [11].

The approach has been experimentally evaluated on a well known benchmark used in the Document Understanding Conference (DUC) to evaluate various proposed algorithms and the results obtained are comparable with some of the best results obtained using other techniques.

The paper is organized as follows. In the next section, we review the GNN model. In Section 3 our approach is described, while Section 4 describes the experimental results on two Document Understanding Conference (DUC) benchmark datasets. Finally, some conclusions are drawn in Section 5.

2 The Graph Neural Network Model

The Graph Neural Network (GNN) is a connectionist model which can accept graphs as inputs. In this paper, instead of giving the formal definition of the

² PageRank was introduced to evaluate the authority of Web pages and was exploited, along with other measures, to sort the URLs returned in response to users' queries in the commercial search engine, Google [9].

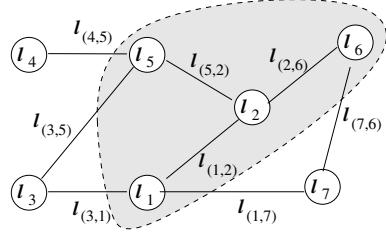


Fig. 1. A graph and the neighborhood of the node with label l_2

model and its training algorithms, for which the readers are referred to [10,11], we will provide an intuitive understanding of the GNN.

In GNNs, a graph represents a set of objects/concepts (the nodes) and their relationships (the edges). For each node n , a *state* $\mathbf{x}_n \in \mathbb{R}^s$ is specified, which automatically stores a representation of the corresponding object. Since every concept is naturally defined by its features and the related concepts, it is assumed that \mathbf{x}_n depends on the information contained in the neighborhood of n (see Fig. 1). Thus, we can use the following model to characterize the state of a node:

$$\mathbf{x}_n = f_{\mathbf{w}}(\mathbf{l}_n, \mathbf{l}_{\text{co}[n]}, \mathbf{x}_{\text{ne}[n]}, \mathbf{l}_{\text{ne}[n]}), \quad (1)$$

where $f_{\mathbf{w}}$ is a parametric function which is called *local transition function*. Moreover, \mathbf{l}_n , $\mathbf{l}_{\text{co}[n]}$, $\mathbf{x}_{\text{ne}[n]}$, $\mathbf{l}_{\text{ne}[n]}$ are respectively the label of n , the labels of its edges, and the states and the labels of the nodes in the neighborhood of n (see Fig. 1). Finally, an output \mathbf{o}_n may also be defined, which depends on the state of the node \mathbf{x}_n and the node label, according to a parametric *local output function* $g_{\mathbf{w}}$:

$$\mathbf{o}_n = g_{\mathbf{w}}(\mathbf{x}_n, \mathbf{l}_n). \quad (2)$$

Thus, Eqs. (1) and (2) specify a parametric model that computes an output $\mathbf{o}_n = \varphi_{\mathbf{w}}(\mathbf{G}, n)$ for any node n of the graph \mathbf{G} . Interestingly, it was proved that, under some mild conditions, a very large class of continuous functions on graphs can be approximated in probability, up to any degree of precision, by the model [11].

The parameters of the unknown model can be estimated using a set of training examples. In practice, the learning can be implemented by the minimization of a quadratic error function

$$e_{\mathbf{w}} = \sum_{i=1}^p \sum_{j=1}^{q_i} (\mathbf{t}_{n_{i,j}} - \varphi_{\mathbf{w}}(\mathbf{G}_i, n_{i,j}))^2, \quad (3)$$

where \mathbf{G}_i is a graph, $n_{i,j}$ one of its nodes, $\mathbf{t}_{n_{i,j}}$ the desired output at that node, p is the number of graphs in the trainset, and q_i is the number of *supervised nodes*³.

³ The supervised nodes are those for which a desired target exists. In general, the supervision can be on all the nodes or on a subset.

In order to implement the GNN model, three issues must be addressed: the implementation of the functions f_w and g_w , the computation of the states \mathbf{x}_u and the computation of the gradient. We will refer the readers to [10][11] to see how these issues can be tackled. Here, it suffices to say that under suitable conditions, f_w and g_w can be implemented using a feedforward neural network. Thus, the parameters can be estimated using a gradient descent algorithm, albeit, the algorithm is more complex than in common neural networks.

3 The Proposed Approach to Sentence Extraction

In our approach, similar to the TextRank method, each document is represented by a graph, where the nodes stand for the sentences and there is an edge between two nodes if the two sentences share some words. Moreover, the edge and the node labels contain features describing the text content (please see Section 3.1 for discussions).

The sentence extraction task can be considered a classification problem, where it has to be predicted, for each sentence in the input document, whether it belongs to a “desired” summary. Thus, in the training phase, the GNN was adapted to learn a function τ such that $\tau(G, n_i) = 1$ if the sentence n_i belongs to the summary, and $\tau(G, n_i) = -1$, otherwise.

During the test phase, the GNN assigns a score (an output) to each sentence and the summary is constructed by selecting those units with the highest scores. In order to decide how many sentences should be included in the summary, two strategies can be adopted: (1) only the units with scores larger than a predefined threshold are selected; or (2) the sentences are sorted according to their scores and the selection can start from the one with the highest score up to when the summary reaches a predefined length. In our experiments, according to the common practice on the chosen benchmarks, the latter strategy has been deployed.

3.1 Node and Edge Label Content

The content of the node and the edge labels depends on the adopted preprocessing steps and on the information available in the considered domain. In our experiments, the documents include a body, a title and a subtitle. Such data can also be considered as minimal information that is usually available in most of the summarization domains. It is worth mentioning, however, that while here we keep the information used by the GNN to a minimal level, an important advantage of the proposed approach lies in the capability that those features can be extended easily.

According to preliminary experimentations and other approaches in literature, the features have been defined as follows: Each edge label contains a score that measures the similarity between the two sentences. Here, we used the same approach adopted by the original TextRank technique [8], where the overlap of

two sentences is defined as the number of common word tokens⁴ divided by a normalization factor, to avoid biasing towards long sentences. More precisely, let $s_k = w_1^k, w_2^k, \dots, w_N^k$, and $k = i, j$, be two sentences and w_p^i, w_q^j be the word tokens in sentence i and j respectively. The similarity $\sigma(s_i, s_j)$ between these two sentences is defined as

$$\sigma(s_i, s_j) = \frac{\#\{w \mid \exists p, q, w = w_p^i = w_q^j\}}{\log(|s_i|) + \log(|s_j|)} \quad (4)$$

where $\#\{\cdot\}$ is the cardinality operator and $|s_k|$ returns the total number of word tokens in s_k .

Four different features have been considered for the node labels: the position of the sentence relative to the beginning of the document, the length of the sentence, the similarity with respect to the title, and the sentence type.

- The position of the sentence relative to the beginning of the document is a positive number that is computed according to the order of the sentences in the text document: it is defined by $\pi(s_i) = \frac{1}{i}$, where $\pi(\cdot)$ stands for the position. Such a feature reflects the well-known fact that the sentences in the first part of the document are more likely to be included in the summaries.
- The length of the sentence is computed by counting the number of its word tokens. $\text{Length}(s_i) = |s_i|$. Longer sentences can contain more information, but they also occupy more space of the total space available for the summary. This feature helps to balance these two factors.
- The similarity value between the sentence and the title is computed as in Eq. (4). The basic idea is that the more words are shared between the title and the sentence, the more important the sentence is. The words in the title are more important, since they are supposed to encapsulate the document content.
- The sentence type represents whether the sentence is the title, the sub-title, or the text in the document and it is represented by the integers 0, 1 and 2 respectively. Sentences that play a different role in the document have a different chance of being included in the summary.

4 Experimental Results

The proposed method has been evaluated on two summarization benchmarks, DUC–2001 [12] and DUC–2002 [13], respectively. These datasets have been chosen, since they allow a comparison with other approaches and they provide the data required for the implementation of the proposed method. The Document Understanding Conference has introduced an evaluation toolkit [14], called ROUGE, to automatically measure the differences between a target summary prepared

⁴ These word tokens are obtained by first de-stemming the word, and removing all common words, and stop words (available from a pre-defined list) from the sentence.

by humans and a summary generated by an algorithm⁵. Thus, several methods, including TextRank, have been applied on DUC–2001 and DUC–2002 and their results, measured using ROUGE, are available. Moreover, the proposed method requires a training dataset where, for each sentence, a target is available specifying whether the sentence should be included in the summary of the corresponding document. To the best of our knowledge, DUC–2001 is the only publicly available dataset that comes with explicit predefined targets for the sentences.

Both DUC–2001 and DUC–2002 have been constructed by the National Institute of Standards and Technology (NIST) including newswires from Wall Street Journal 1987–1992, AP newswire 1989–1990, San Jose Mercury News 1991, Financial Times 1991–1994, LA Times, and FBIS. The documents are grouped in sets, with an average of 10 per set, each one discussing the same topic. The documents are at least 10 sentences long and there is no maximum length. DUC–2001 contains 60 sets of documents (30 for training and 30 for testing), while DUC–2002 contains 59 sets. The dataset guidelines specify that DUC–2002 documents should be used only for testing purposes, whereas the training has to be carried out on the DUC–2001 training set.

In our experiments, the task of producing a 100 word summary has been considered. DUC–2001 and DUC–2002 include two example summaries for each document. The summaries have been originally generated by humans and they do not directly correspond to any set of sentences. However, a set of extracts for the DUC–2001 training set has been included later. These extracts have been manually constructed in an attempt to find the extracts which are closer to the original human generated summary.

All the results in this paper have been obtained by a GNN architecture, where the state dimension was set to 2 and both the transition and the output functions were implemented by a layered neural networks with one hidden layer with 5 hidden neurons. The training was run for 1000 epochs. These parameters have been selected by a trial and error procedure carried out on the training set.

4.1 Evaluation of the Results

The ROUGE tool computes measures to automatically evaluate the quality of a summary in comparison with other summaries created by humans. This tool has been experimentally evaluated in [14] and it has been shown that the judgments obtained by the proposed measures are highly correlated with those made by humans.

The ROUGE measures basically count the number of overlapping units such as n-grams, word sequences and word pairs between the computer generated

⁵ Obviously, ROUGE cannot completely replace the human judgment which is able to evaluate more carefully the quality of a summary. However, a fair human comparison of different approaches is resource intensive, since it requires the organization of a competition with a formal review procedure. On the other hand, it has been experimentally shown that ROUGE judgments are highly correlated with those made by humans [14].

summary, and the ideal summary created by humans. In our experiments, we used four of the most significant ROUGE measures: Rouge k, Rouge-su k and and Rouge-w h.k. Rouge n is a measure that depends on the number of k pairs of words shared by two summaries. This measurement does not take in account the order of the words that, on the other hand, is considered by Rouge-su k, which depends on the common sub-sequences of words up the length of k . Moreover, Rouge-su h.k is a more complex measure that counts the sequences up to the length k and with the possibility of skipping up to h words. Finally, for each measure, ROUGE returns three values, i.e., precision, recall and f-measure, that corresponds to the common performance test in information retrieval. The measures are also often further specialized by allowing ROUGE to remove a set of common words (stop words) and/or to reduce inflected words to their stems (stemming). More details can be found in [14].

A preliminary set of experiments was used to evaluate the impact of each node feature on the performance of the predictor. Thus, a GNN was trained to predict the importance of the sentences using only one kind of node feature in each run. Table 1 shows the values of Rouge 1 (R-1), Rouge 2 (R-2), Rouge-su 4 (R-SU-4) and and Rouge-w 1.2 (R-W-1.2) achieved on DUC-2001⁶. The performances on DUC-2002 are similar. Interestingly, the results suggest that, even if all the labels may contribute the final output, however the sentence position is the most informative feature. Such a fact confirms the intuitive expectation that most important sentences are written at the beginning of the documents.

Table 1. A comparison on DUC2001 of the impact of each feature on the performance

Label node	DUC-2001							
	R-1		R-2		R-SU-4		R-W-1.2	
Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	
Sentence Position	0.4330	0.4287	0.1922	0.1906	0.2100	0.2079	0.2326	0.0911
Number Words	0.4103	0.4051	0.1641	0.1622	0.1878	0.1855	0.2095	0.0818
Sentence Type	0.4111	0.4079	0.1672	0.1662	0.1886	0.1872	0.2129	0.0836
Relation with Title	0.4205	0.4164	0.1760	0.1745	0.1969	0.1950	0.2193	0.0859

Then, the proposed approach has been applied, using all the features, on both DUC-2001 and DUC-2002 and it has been compared with TexRank, a baseline⁷, and the competitors' submissions to the competition In Table 2 and [3], respectively, we report only the best results reached during the

⁶ In the present experimentation, the parameter of the measures have been chosen following the original ROUGE assessment [14].

⁷ The baseline summaries have been constructed using the first sentences of each documents until 100 words are reached.

competitions⁸. The tables show the performance obtained comparing the original summaries (CASE), the stemmed version of the summaries (STEM), and the stopped version of the summaries (STOP).

Table 2. A comparison of the results on DUC–2001. The best results of the participant to the competition, which did not include TexRank and GNN, are in gray.

Method	PRECISION														
	R-1			R-2			R-SU-4			R-W-1.2					
CASE STEM STOP				CASE STEM STOP				CASE STEM STOP				CASE STEM STOP			
Baseline	0.2680	0.2795	0.2129	0.1187	0.1220	0.1023	0.1283	0.1342	0.0992	0.1471	0.1510	0.1408			
O	0.2579	0.2700	0.2052	0.1070	0.1103	0.0919	0.1192	0.1253	0.0924	0.1360	0.1401	0.1306			
T	0.2805	0.2929	0.2245	0.1260	0.1298	0.1081	0.1358	0.1423	0.1060	0.1524	0.1567	0.1447			
V	0.2841	0.2963	0.2257	0.1237	0.1274	0.1076	0.1352	0.1415	0.1054	0.1569	0.1609	0.1487			
TextRank	0.4133	0.4342	0.3642	0.1694	0.1757	0.1602	0.1910	0.2021	0.1688	0.2139	0.2205	0.2259			
GNN	0.4251	0.4448	0.3763	0.1872	0.1928	0.1801	0.2058	0.2161	0.1831	0.2278	0.2343	0.2454			
RECALL															
Baseline	0.4370	0.4556	0.3778	0.1935	0.1989	0.1825	0.2107	0.2203	0.1792	0.0948	0.0973	0.1114			
O	0.4487	0.4697	0.3854	0.1874	0.1932	0.1742	0.2099	0.2206	0.1774	0.0937	0.0965	0.1094			
T	0.4449	0.4645	0.3950	0.2013	0.2074	0.1920	0.2179	0.2284	0.1907	0.0957	0.0984	0.1135			
V	0.3791	0.3954	0.3311	0.1644	0.1692	0.1579	0.1807	0.1892	0.1564	0.0826	0.0848	0.0972			
TextRank	0.4094	0.4301	0.3643	0.1680	0.1742	0.1609	0.1892	0.2002	0.1694	0.0838	0.0864	0.1008			
GNN	0.4196	0.4391	0.3613	0.1850	0.1905	0.1738	0.2032	0.2133	0.1762	0.0889	0.0915	0.1050			
F-MEASURE															
Baseline	0.3317	0.3459	0.2715	0.1469	0.1510	0.1307	0.1592	0.1666	0.1273	0.1151	0.1182	0.1239			
O	0.3263	0.3415	0.2666	0.1358	0.1400	0.1199	0.1515	0.1593	0.1209	0.1105	0.1139	0.1184			
T	0.3431	0.3582	0.2851	0.1545	0.1592	0.1377	0.1669	0.1748	0.1357	0.1172	0.1205	0.1266			
V	0.3237	0.3376	0.2672	0.1407	0.1449	0.1274	0.1541	0.1613	0.1254	0.1079	0.1107	0.1170			
TextRank	0.4133	0.4320	0.3631	0.1694	0.1749	0.1601	0.1910	0.2011	0.1686	0.1139	0.1241	0.1390			
GNN	0.4222	0.4418	0.3676	0.1861	0.1916	0.1764	0.2044	0.2146	0.179	0.1279	0.1315	0.1467			

It is observed, in the tables, that the approach proposed in this paper performs better than any others in terms of precision and F-score, while the recall score is comparable to the best results of other approaches. Moreover, our method always outperforms TextRank, confirming that the GNNs are able to model the original TexRank behavior and to improve its results.

⁸ A detailed description of the methods, denoted by O, T, V, 23, 27 and 28, can be found in the proceedings of the DUC 2001 and DUC 2002 Workshops on Text Summarization.

Table 3. ROUGE result for participants to DUC-2002. The best results of the participant to the competition, which did not include TexRank and GNN, are in gray.

PRECISION											
	R-1	R-2	R-SU-4	R-W-1.2							
Method	CASE STEM STOP	CASE STEM STOP	CASE STEM STOP	CASE STEM STOP							
23	0.3102 0.3221 0.2532	0.1443 0.1482 0.1261	0.1527 0.1592 0.1219	0.1693 0.1736 0.1653							
27	0.2837 0.2953 0.2360	0.1310 0.1349 0.1147	0.1390 0.1453 0.1118	0.1525 0.1570 0.1505							
28	0.3126 0.3251 0.2611	0.1470 0.1512 0.1292	0.1551 0.1620 0.1262	0.1664 0.1708 0.1648							
GNN	0.4606 0.4788 0.4235	0.2157 0.2214 0.2110	0.2294 0.2395 0.2109	0.2475 0.2539 0.2734							
RECALL											
23	0.4146 0.4306 0.3746	0.1938 0.1990 0.1879	0.2056 0.2143 0.1829	0.0895 0.0918 0.1087							
27	0.4891 0.5092 0.4464	0.2263 0.2330 0.2178	0.2416 0.2528 0.2154	0.1038 0.1068 0.1266							
28	0.4771 0.4961 0.4389	0.2250 0.2314 0.2187	0.2387 0.2493 0.2160	0.1006 0.1032 0.1235							
GNN	0.4534 0.4714 0.4085	0.2124 0.2180 0.2040	0.2258 0.2358 0.2036	0.0963 0.0988 0.1173							
F-MEASURE											
23	0.3529 0.3665 0.3003	0.1645 0.1690 0.1500	0.1742 0.1816 0.1453	0.1165 0.1195 0.1303							
27	0.3533 0.3677 0.3040	0.1634 0.1682 0.1481	0.1736 0.1816 0.1450	0.1214 0.1249 0.1351							
28	0.3764 0.3914 0.3258	0.1772 0.1822 0.1617	0.1873 0.1957 0.1585	0.1249 0.1283 0.1404							
GNN	0.4568 0.4749 0.4148	0.2139 0.2196 0.2069	0.2275 0.2375 0.2066	0.1386 0.1422 0.1638							

5 Conclusions

In this paper, we have applied a recently introduced connectionist model, namely, Graph Neural Network (GNN), on the sentence extraction problem. We evaluated the approach on a common benchmark and we found that the results compares very well with the best performances of other approaches.

One of the advantages of our approach is that it is completely data-driven. In other words, once the parameters of the GNN are fixed, through some initial experimentation, the approach is completely driven by the data, and there is very little human intervention required. Those results are promising and suggest that the method can be used in real life text summarization applications. It is matter of further research a wider experimentation and the evaluation of more sources of information to be used as features in the text graph. More generally, text summarization is a very difficult task and its solution with human performance will probably require the use and the integration of complex syntactic and semantic domain information.

Acknowledgment

The authors wish to acknowledge the partial financial support received from the Australian Research Council, Discovery Project grant DP0774168 and Linkage International Award grant LX0882106, which made an extended stay of the first author in University of Wollongong possible.

References

1. Mani, I.: Automatic summarization. *Computational Linguistics* 28(2) (2001)
2. Marcu, D.: Improving summarization through rhetorical parsing tuning. In: The 6th Workshop on Very Large Corpora, pp. 206–215 (1998)
3. Barzilay, R., Elhadad, M.: Using lexical chains for text summarization. In: Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, vol. 17 (1997)
4. Larsen, B.: A trainable summarizer with knowledge acquired from robust NLP techniques. In: Advances in Automatic Text Summarization, p. 71 (1999)
5. Lin, C.: Training a selection function for extraction. In: Proceedings of the eighth international conference on Information and knowledge management, pp. 55–62. ACM, New York (1999)
6. Svore, K., Vanderwende, L., Burges, C.: Enhancing single-document summarization by combining RankNet and third-party sources. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 448–457 (2007)
7. Conroy, J., O'leary, D.: Text summarization via hidden markov models. In: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, p. 407. ACM, New York (2001)
8. Mihalcea, R., Tarau, P.: TextRank: Bringing order into texts. In: Proceedings of EMNLP, pp. 404–411. ACL, Barcelona (2004)
9. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web (1998)
10. Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., Monfardini, G.: The graph neural network model. *IEEE Transactions on Neural Networks* 20(1), 61–80 (2009)
11. Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., Monfardini, G.: Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks* 20(1), 81–102 (2009)
12. Over, P.: Introduction to DUC-2001: an intrinsic evaluation of generic news text summarization systems. In: ACM SIGIR 2001 Workshop on Automatic Summarization (2001)
13. Over, P., Liggett, W., Gilbert, H., Sakharov, A., Thatcher, M.: Introduction to DUC-2002: An intrinsic evaluation of generic news text summarization systems. In: Workshop on Automatic Summarization (2002)
14. Lin, C.: Rouge: A package for automatic evaluation of summaries. In: Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), pp. 25–26 (2004)

Autonomous Generation of Internal Representations for Associative Learning

Michaël Garcia Ortiz, Benjamin Dittes,
Jannik Fritsch, and Alexander Gepperth

Honda Research Institute Europe GmbH
Carl-Legien-Str.30, 63073 Offenbach,
Germany

CoR-Lab, Bielefeld University
Universitätsstr. 25, 33615 Bielefeld,
Germany

michael.garcia.ortiz@gmail.com,
{benjamin.dittes,jannik.fritsch,alexander.gepperth}@honda-ri.de

Abstract. In this contribution, we explore the possibilities of learning in large-scale, multimodal processing systems operating under real-world conditions. Using an instance of a large-scale object detection system for complex traffic scenes, we demonstrate that there is a great deal of very robust correlations between high-level processing results quantities, and that such correlations can be autonomously detected and exploited to improve performance. We formulate requirements for performing *system-level learning* (online operation, scalability to high-dimensional inputs, data mining ability, generality and simplicity) and present a suitable neural learning strategy. We apply this method to infer the identity of objects from multimodal object properties (“context”) computed within the correlated system and demonstrate strong performance improvements as well as significant generalization. Finally, we compare our approach to state-of-the-art learning methods, Locally Weighted Projection Regression (LWPR) and Multilayer Perceptron (MLP), and discuss the results in terms of the requirements for system-level learning.

1 Introduction

In contrast to many previous approaches focussing on learning close to sensory signals [1][2], this contribution supports an alternative view, namely that learning in large-scale environment perception systems gets more feasible and beneficial when applied to heavily preprocessed, abstract/invariant quantites to which we will summarily refer to as *system-level quantities*. We propose that this kind of *system-level learning* has, by construction, a high generalization ability which supports system performance particularly in difficult sensory conditions (high-dimensionality, noise, ambiguity, ...).

This study is conducted based on an instance of a large-scale object detection system in road traffic environments [3] which integrates multimodal information

(laser, video) as well as a wide variety of vision-based algorithms such as stereo, tracking, classification, and free-area detection. The motivation for this study arose when trying to obtain multimodal object models (in this case: car models) for excluding obviously incorrect object detections.

1.1 Requirements for Learning Strategies

For being applicable for system-level learning in a large-scale system, any learning method must fulfill several requirements:

Online regression, i.e., the ability to train and perform regression (in contrast to binary classification) within a running system, which implies that the number of training examples is not known in advance, while avoiding catastrophic forgetting [4] when new examples are presented to an already trained system.

Generality and simplicity, a requirement which concerns the applicability of an algorithm to a very wide range of data. Thus, e.g., support vector machines with problem-specific kernels such as, e.g. [5], are inapplicable. Any system-level learning method must also be *simple* in the sense that it does not contain a large number of parameters that must be tuned to problem-specific values.

Scalability. Using a simple set of system-level learning methods requires the conversion of system-level quantities into a common representational format (see [6]). Such a format should not be optimized for a specific kind of data, and should give the possibility of representing probabilistic information. Therefore, any system-level learning algorithm must cope with the fact that the number of data dimensions can grow very large.

Data mining ability. A learning algorithm must be able to ignore irrelevant and possibly noisy dimensions and to approximately identify the relevant ones, especially in high-dimensional data.

Reusability of internal representations. Any advanced learning algorithm constructs internal representations of the input data, e.g. the hidden layers of a MLP or the set of support vectors of a support vector machine. Internal representations should be reusable for other tasks if possible.

1.2 Neural Projection-Prediction as a System-Level Learning Strategy

We present a first instance of an extensible, composite neural method able to exploit correlations and interrelationships between system-level quantities, especially higher-order correlations not directly available to direct associative learning. This contribution focuses on the autonomous generation of internal representations by the proposed *neural projection-prediction* (NPP) method. We demonstrate the benefits and limitations of our composite method by rigorously benchmarking it. For this purpose, we employ data in the form of system-level quantities recorded from a large-scale real-world system dedicated to object detection in challenging traffic environments. In order to realistically assess the

value of NPP as a system-level learning method, we evaluate two other learning methods on the benchmark task: a nonlinear MLP and the LWPR [7].

1.3 Related Work

LWPR [7] is explicitly designed to be an online method for high-dimensional data, and it avoids catastrophic forgetting by incrementally partitioning the input space into volumes where individual linear regression models are applied. Here, we evaluate the performance of LWPR w.r.t. suitability for system-level learning. We particularly intend to determine whether LWPR can deal with the requirements of scalability and generality (sec. I.I).

Another related model is the Radial Basis Function network (RBFN) model [4] which filters the input data through a set of Radial Basis Functions (RBFs) performing linear regression. The RBF centers and widths are chosen by the user. This generates a large number of parameters, which violates the requirements of generality, simplicity and reusability (Sec. I.I). We therefore consider the RBFN model to be incompatible with our requirements.

The multilayer perceptron (MLP) model trained by a batch-mode back-propagation algorithm (see, e.g., [84]) fulfills many of the requirements specified in Sec. I, although the online property and the reusability of internal representations are not guaranteed (since the back-propagation mechanism tunes the hidden layer exclusively w.r.t. the learning task). We will study this model in order to assess its potential as a system-level learning algorithm.

2 Methods

We used population coding in our system to realize a common representational format (see [6] for details) which allows to use learning methods fulfilling the requirement of scalability presented in Sec. I.I. A system-level quantity is represented by an activation on a two-dimensional surface, where the position and amplitude of the activation code for the (possibly two-dimensional) value and the confidence, respectively (see Fig. II for examples). Simple quantities (distance, height, size, ..) can be represented naturally in this way; information of higher intrinsic dimensionality must be subjected to a suitable *projection*, which will be elaborated in more detail in Sec. 2.2. This way of storing information is not optimized for storage efficiency, which leads to high-dimensional data even though the intrinsic dimensionality of represented quantities is low.

2.1 Notation

We denote local activity in a two-dimensional population code A by $z^A(\mathbf{x}, t)$. A neuron at position \mathbf{x} of population-code A is connected to another neuron at position \mathbf{y} of population-code B by a weight matrix denoted $w_{\mathbf{x}\mathbf{y}}^{AB}$.

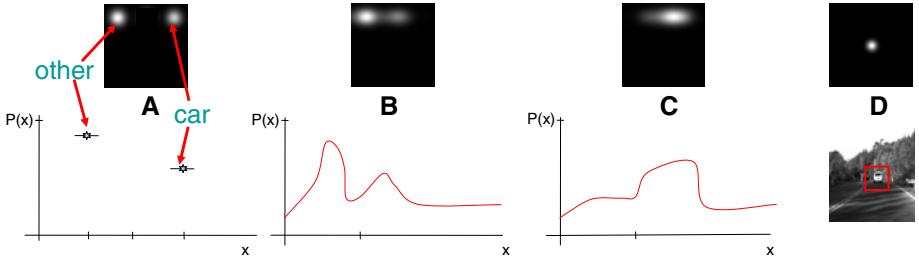


Fig. 1. Examples of system-level quantities from the benchmark dataset. A) The discrete distribution from an object classifier is translated into a population code where only certain locations carry information. B),C) Quasi-continuous one-dimensional measurements (here: object elevation and distance) are encoded into population codes that are extended along one axis. The uncertainty (multimodality) of measured distributions is transferred to the resulting population code. D) Quasi-continuous two-dimensional measurements (here: object position in camera image) are naturally encoded into a two-dimensional population code.

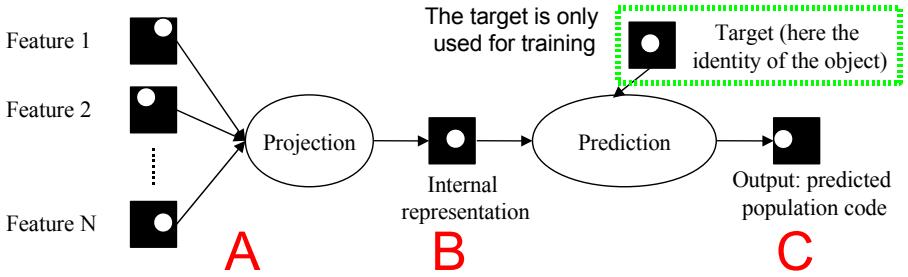


Fig. 2. Description of the system

2.2 Neural Projection-Prediction

The technique we present consists of two steps: a self-organizing map (SOM) [9] performing a *projection* of the input space on a two-dimensional *internal representation*, and a neural network mapping the internal representation to a population-coded output using a normalized form of logistic regression. NPP is an online method: adaptation of the SOM and the logistic regression weights are performed in parallel to data transmission. The weight vectors of both projection and prediction are initialized with small random values.

Neural Projection-Prediction: Projection step. In the projection step, the input A consisting of a concatenation of population codes is projected onto a two-dimensional output B using a self-organizing map (SOM, see [9]). At the same time, the SOM weights are updated according to the conventional rule based on input activity $z^A(\mathbf{x}, t)$, output activity $z^B(\mathbf{y}, t)$ and the position of the best-matching unit, c . Projection and updating are governed by:

$$\begin{aligned}
z^B(\mathbf{y}, t) &= \sum_{\mathbf{x}} w_{\mathbf{x}\mathbf{y}}^{AB} z^A(\mathbf{x}, t) \\
w_{\mathbf{x}\mathbf{y}}^{AB}(t+1) &= w_{\mathbf{x}\mathbf{y}}^{AB}(t) + \epsilon_{\text{proj}} g_c^\sigma(\mathbf{y}) [w_{\mathbf{x}\mathbf{y}}^{AB}(t) - z^A(\mathbf{x}, t)] \\
\text{where } g_c^\sigma(\mathbf{y}) &= \exp - \frac{\mathbf{y} - c}{2\sigma^2}.
\end{aligned} \tag{1}$$

The values of ϵ_{proj} and σ are usually decreased over time. The precise form of this variation can influence projection quality considerably.

Neural Projection-Prediction: Prediction step. We employ a supervised learning strategy where the supervision signal can come from annotated data or can be generated within the system (bootstrapping). Based on activity $z^B(\mathbf{x}, t)$ in the internal representation B , we use logistic regression [10] for training, minimizing the error of the prediction for C , $z^C(\mathbf{y}, t)$, based on the teaching signal $t^C(\mathbf{y}, t)$. The transmission and learning rules for the prediction step read:

$$\begin{aligned}
z^C(\mathbf{y}, t) &= \sigma \left(\sum_{\mathbf{x}} w_{\mathbf{x}\mathbf{y}}^{BC} z^B(\mathbf{x}, t) \right) \text{ with } \sigma(x) \equiv \frac{1}{1 + \exp(\mu - x)} \\
w_{\mathbf{x}\mathbf{y}}^{BC}(t+1) &= w_{\mathbf{x}\mathbf{y}}^{BC}(t) + \epsilon_{\text{pred}} z^B(\mathbf{x}) [z^C(\mathbf{y}, t) - t^C(\mathbf{y}, t)]
\end{aligned} \tag{2}$$

where σ is correcting its argument by subtracting the long-term mean μ .

2.3 Baseline Techniques

The following techniques serve as a baseline to demonstrate the advantages of NPP and the possible ways to improve it.

Locally weighted projection regression. LWPR is a method for learning high-dimensional function approximators based on the superposition of multiple linear models in the input space. We used the publicly available implementation of LWPR [7] by the authors for all described experiments. Since LWPR stores a covariance matrix for each used linear model, it cannot deal with very high-dimensional data of $d > 1000$ due to memory consumption.

The multilayer perceptron model. The MLP model [4] is a standard non-parametric regression method using gradient-based learning. It is a rather simple model, the only real free parameters being the number and size of hidden layers. The hidden layer may be viewed as an abstract internal representation where it is however unclear what is being represented. For network training, we employ the backpropagation algorithm with weight-decay and a momentum term (see, e.g., [8]). We used the pyBrain-library [11] for all described MLP experiments.

3 Experiments and Results

The *standard dataset* consists of 30000 samples of system-level quantities recorded in a large-scale integrated object detection system. A car is equipped with two

front cameras and a laser in order to acquire data which is used by several processing layers [3]. The output of these layers is converted into population-codes (PCs) following the method described in Sec 2. Each sample of the *standard dataset* consists of 8 population-coded quantities representing abstract, invariant visual and spatial properties of objects detected by the system: distance, size, image position, elevation, two measures for distance-to-road-area, retinal size and depth (see [6]). Each PC consists of 64x64 elements, one sample therefore has a dimensionality of 32768. For training purposes, each PC is downsampled to a size of 16x16, making the dimensionality of one example $d = 2048$. The *reduced dataset* contains three PC per sample: retinal XY position, retinal size and distance. The *noised dataset* contains the 8 PC, plus 5 PC of uniform noise. As explained in Sec 2, the effective dimensionality is lower than the size of the PC, because some areas of the PC are not used. However, we keep the unnecessary dimensions to show that NPP is robust under the addition of irrelevant dimensions (*data mining ability*, see Sec. 1.1).

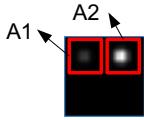
We generate several *LWPR datasets* of roughly the same content but reduced dimensionality since LWPR cannot deal with the high dimensionality of the default dataset. This is achieved by computing the coordinates (2 numbers) of the center of gravity for each of the 8 PCs contained in a single data sample. The dataset LWPR-1 is the approximation of the *reduced dataset* and has 4 dimensions since image position carries two-dimensional information. Dataset LWPR-2 has dimensionality 9 approximating the 8 PCs (in most cases the y-coordinate of the center of gravity can be disregarded), and Dataset LWPR-3 has dimensionality 18, where 9 dimensions are taken from LWPR-2 and 9 dimensions contain uniform noise.

The examples are either positive (coming from car objects) or negative (not coming from cars). The ratio of positive to negative examples is approximately 1:10. For all training runs, we split the used dataset: the first 15000 examples are used for training, the last 15000 examples for evaluation.

3.1 Evaluation Measures

The output of the trained Neural Projection-Prediction architecture is a discrete population code $z^C(\mathbf{x}, t)$ predicting the identity of a detected object (see Fig. II). The population code is formed by two regions of activations, a_1 coding for the class "car" and a_2 for the class "non-car". We calculate a confidence value $C = A_1 - A_2$ with $A_{1,2} = \sum_{a_{1,2}} z^C(\mathbf{x}, t)$ for each example. A decision is made by comparing C to a variable threshold θ . This is visualized in Fig. 3.

This decision can be compared to the known "true" class of the object given by the identity. For different values of θ , we calculate the *false positive rate* $fpr = \frac{\#(\text{incorrect positive classifications})}{\#(\text{negative examples})}$ and the *false negative rate* $fnr = \frac{\#(\text{incorrect negative classifications})}{\#(\text{positive examples})}$ for the whole stream. We will assess classification quality by fnr-fpr plots also known as *receiver-operator characteristics* (ROCs).



$$C = A_1 - A_2$$

$$\text{class} = \begin{cases} \text{"car(positive)"} & C > \theta \\ \text{"non-car(negative)"} & \text{otherwise} \end{cases}$$

Fig. 3. Decision making process for performance evaluation

3.2 Experiments with Baseline Techniques

We evaluated LWPR using the datasets LWPR-1, LWPR-2, LWPR-3. Default LWPR parameters were used, except for an initial distance metric of 0.0625 and enabled meta learning. By changing the decision threshold θ applied to the real-valued output of LWPR, ROCs were produced which can be seen in Fig. 4. The figure shows that LWPR is able to solve the classification task well and improves greatly by adding more input dimensions. However, it has difficulty coping with unnecessary dimensions. Furthermore, LWPR training failed due to memory limitations when using the *default dataset* or the *reduced dataset* due to the number of receptive fields. It is not suitable to work with such high dimensionalities in its present form. It takes approximately 4 *rounds* of data for the LWPR to converge (one round is one iteration over the whole dataset). It creates 170 receptive fields for LWPR-1, 790 for LWPR-2, and 730 for LWPR-3.

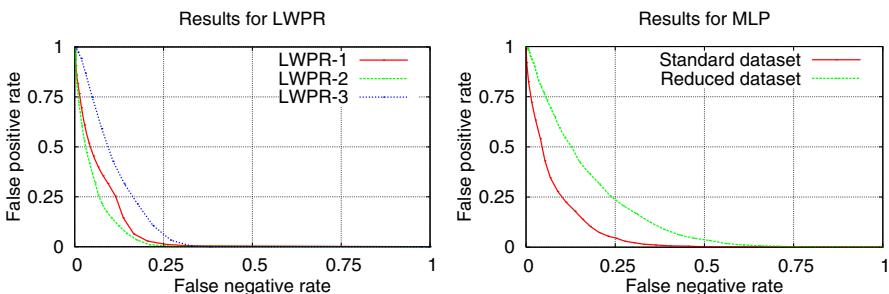


Fig. 4. Results for LWPR and MLP algorithms

We trained an MLP as described in Sec. 2.3 using the standard and the reduced dataset (see Sec. 3). We employ MLP networks with an input layer of size 2560, one hidden layer of size 50 and one output neuron, using a sigmoid nonlinearity for each neuron. We verified that the results are similar for a number of hidden units between 50 and 100 hidden units. Training of the MLP requires 5 rounds (gradient steps) before early-stopping occurs. Training convergence was fast in spite of the high input dimensionality, resulting in ROCs given in Fig. 4 by applying a varying threshold θ to the real-valued MLP output.

3.3 Experiments with Neural Projection-Prediction

We trained our algorithm on the *standard*, *reduced* and *noised datasets*, and we limit the training to one round of data. We impose this constraint to our algorithm in order to evaluate its potential on online scenarios.

Reduced Dataset. We use a standard value for the learning constant of the prediction: $\epsilon_{pred} \simeq 1/15000$, 15000 being the number of training examples. We verified during our experiments that the value of the learning constant of the projection can vary between 0.1 and 0.0001 without affecting the results. We also performed experiments with different values and decreasing functions for the SOM radius. SOMs are usually trained with a decreasing radius, which implies a fixed number of training samples and a converging network. As we aim to work on real-world online problem, we verify that we obtain comparable results for a fixed radius. One can observe on Fig. 5, for $\epsilon_{proj} = 0.001$ and $\epsilon_{pred} = 0.0001$, that having a fixed radius does not change the results of the prediction drastically.

Performance with a high dimensionality. We now run our algorithm on the *standard dataset*. Our algorithm requires a very small amount of parameter tuning, as it performs similarly for a large span of values. We keep a constant radius $\sigma = 10$. One can observe on Fig. 5 that the performances with the *standard dataset* are better than the performances with the *reduced dataset*, especially in the high false-negative rate regime.

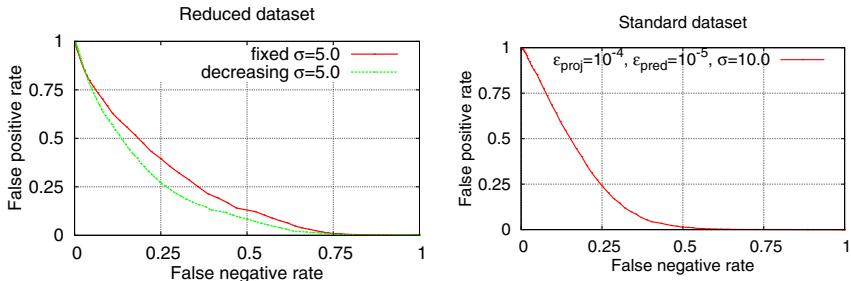


Fig. 5. We can observe on the left figure the results for a fixed radius and a linearly decreasing radius (from 5.0 to 1.0), using the reduced dataset. On the right we show the result for the standard dataset, using a fixed radius.

We show with this experiment that the NPP complies with several requirements from Sec. II: simplicity and scalability. As the SOM projection does not depend on the target value, the internal representations are not task-specific, and thus reusable. The technique is less efficient than the LWPR or the MLP, which have a task-specific internal representation. This lower performance is the price to pay for the reusability of the internal representation. Another reason for the lower performance is the fact that we perform only one round of learning.

Resistance to noise. Fig. 6 shows the results of the experiments conducted in order to study the performance of our algorithm with noisy data. We used the *Noised dataset* which has a dimensionality of 3328, with standard parameter values taken from the previous section. We can observe that the noise does not affect the NPP. As expected, the SOM algorithm is resistant to noisy dimensions.

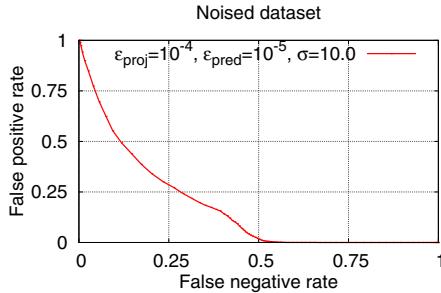


Fig. 6. Analysis of the resistance to noise

4 Conclusion and Key Findings

Based on data obtained from a large-scale processing system, we study three learning techniques to determine whether they fulfill the requirements described in Sec. 1.1 related to system-level learning in large scale architectures or not.

The LWPR algorithm gives the best results regarding the quality of the prediction. It also prevents catastrophic forgetting, and so fulfills the requirement of *online regression*. However, the internal representation is task-specific, it is unable to cope with too high dimensions or with additional noise, so it does not meet several requirements: *scalability*, *data mining ability*, and *reusability of internal representations*. Also, we observe that it performs best with a certain amount of manual tuning (selection of input dimensions and parametrization of distance metrics), which goes against the requirement of *simplicity*. LWPR is then not suitable for our system-level learning requirements.

The MLP algorithm also performs well in terms of the quality of the prediction. The internal representation (hidden layer) is not easily reusable since it is strongly influenced by the chosen learning task. The MLP algorithm meets the criteria of *scalability*, *data mining* and obviously *generality and simplicity*, since the internal representation size is found to be uncritical. However it is not an online algorithm and it faces known issues of catastrophic forgetting (see [8]), and so it is unsuitable for an online real-world system, where new combinaisons of inputs can overwrite previously learned combinaisons.

The NPP meets all the requirements described in Sec. 1.1. It is a *generic and simple* method with a small amount of parameters, and usable in a plug and play manner. It is able to handle very high dimensions (*scalability*), and is not disturbed by unnecessary dimensions (*data mining ability*). The internal

representation does not depend on the task, only on the input space (*reusability of internal representations*). Finally, it performs *online regression* where LWPR and MLP need to use the training data several times. We want to emphasize the fact that our goal is the system-level applicability. As we have shown, we pay the price for this in the form of reduced performance. Even so, the benefits for the whole system can be huge, especially in real-world applications.

5 Future Works

As future research topics, we propose several additional mechanisms. We want to increase the performance while retaining the advantages we gained. We will first improve the projection in order to represent the input space as faithfully as possible. We also plan to derive a feedback signal in order to modulate the SOM clustering depending on the quality of the prediction.

References

1. Gepperth, A., Mersch, B., Fritsch, J., Goerick, C.: Color object recognition in real-world scenes. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4669. Springer, Heidelberg (2007)
2. Wersing, H., Körner, E.: Learning optimized features for hierarchical models of invariant object recognition. *Neural Computation* 15(7) (2003)
3. Schmüderich, J., Einecke, N., Hasler, S., Gepperth, A., Bolder, B., Rebhan, S., Franzius, M., Kastner, R., Dittes, B., Wersing, H.: System approach for multi-purpose representations of traffic scene elements. In: Proceedings of the ITSC (2010)
4. Haykin, S.: Neural networks: a comprehensive foundation. Prentice-Hall, Englewood Cliffs (1999)
5. Mersch, B., Glasmachers, T., Meinicke, P., Igel, C.: Evolutionary optimization of sequence kernels for detection of bacterial gene starts. *Int. J. Neural. Syst.* 17(5) (2007)
6. Gepperth, A., Fritsch, J., Goerick, C.: Cross-module learning as a first step towards a cognitive system concept. In: Proceedings of the First International Conference On Cognitive Systems (2008)
7. Vijayakumar, S., Schaal, S.: Locally weighted projection regression: Incremental real time learning in high dimensional space. In: ICML 2000: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 1079–1086. Morgan Kaufmann Publishers Inc., San Francisco (2000)
8. Reed, R.J., Marks II, R.J.: Neural smithing: Supervised Learning in Feedforward Artificial Neural Networks. MIT Press, Cambridge (1999)
9. Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biol. Cybernet.* 43, 59–69 (1982)
10. Bishop: Pattern recognition and machine learning. Springer, Heidelberg (2006)
11. Schaul, T., Bayer, J., Wierstra, D., Yi, S., Felder, M., Sehnke, F., Rückstieß, T., Schmidhuber, J.: Pybrain. *Journal of Machine Learning Research* (2010)

Improving Accuracy of LVQ Algorithm by Instance Weighting

Marcin Blachnik¹ and Włodzisław Duch²

¹ Department of Management and Informatics, Silesian University of Technology
Katowice, Krasinskiego 8, Poland
marcin.blachnik@polsl.pl

² Department of Informatics, Nicolaus Copernicus University,
Grudziądzka 5, Toruń, Poland; Google: W Duch

Abstract. Similarity-based methods belong to the most accurate data mining approaches. A large group of such methods is based on instance selection and optimization, with Learning Vector Quantization (LVQ) algorithm being a prominent example. Accuracy of LVQ highly depends on proper initialization of prototypes and the optimization mechanism. Prototype initialization based on context dependent clustering is introduced, and modification of the LVQ cost function that utilizes additional information about class-dependent distribution of training vectors. The new method is illustrated on 6 benchmark datasets, finding simple and accurate models of data in form of prototype-based rules.

1 Introduction

Lazy learning has many advantages [1] and can be extended in many ways [2][3]. Nearest neighbor algorithms belong to the simplest models of the similarity-based learning (SBL) framework. They need very little training (parameter selection), are frequently quite accurate, may use any type of attributes, and may be applied directly to similarity matrices in cases when vector-based representation of information is not possible. They may use specialized kernels [4] for similarity evaluation, including kernels for time series, biological sequences, spectral kernels, graphs and other complex structures, mutual information and other measures of dependency etc. [5][6]. Although these kernels are commonly used in Support Vector Machines (SVMs) to create extended feature space in which linear discrimination is performed they may also be used in any SBL algorithm. This is still a relatively unexplored area in machine learning.

Unfortunately similarity-based algorithms, such as the k-nearest neighbor (kNN) method, also have some drawbacks. First, although there is little learning prediction may be time-consuming, because it requires distance calculations between the test t_i and all training set instances $D = [d_1, d_2, \dots, d_n]$. Several algorithms for exact and approximate fast computation of distance exist. For example, BallTree or KDTree [7] algorithms reduce complexity of distance calculations by using tree-structured search methods (see recent paper [8] for review of approximate ϵ -NN algorithms). Another approach that can reduce prediction time, improve prediction accuracy and comprehensibility of the kNN algorithm is based on selection and optimization of the training set reference vectors D .

Selection removes those instances from the training data that do not bring significant gains in the prediction process. Some examples fo algorithms that belong to this group include top-down algorithms: **Edited Nearest Neighbor (ENN)** [9] removes from the training set instances that are incorrectly classified by kNN for $k \geq 3$; **DROP3** [10] removes instance x from the training set if it does not change classification of instances for which x is one of the k nearest neighbors; **Iterative Case Filtering (ICF)** [11] starts from DROP3 and creates hyperspheres with instances from a single-class. Bottom-up algorithms: **CNN (Condensed Nearest Neighbor rule)** starts with one vector per class and adds training instances that are wrongly classified by the current reference set. **Graph-based methods:** **Gabriel Editing (GE)** [12] uses Gabriel graph to remove from the training dataset all instances from the same class as all their neighbors, leaving only those instances that define decision borders between different classes. Probability-density estimating algorithms: **Edited NRBF** [13] uses normalized RBF to remove instances x inconsistent with probability density estimation around point x .

Optimization methods may start from the selected vectors but shift reference instances to optimal positions. Typical examples that belong to optimization group are numerous clustering algorithms, including the LVQ algorithm [14][15], which is used in this paper.

Empirical comparison of these methods [13][16] shows that the best results are usually achieved by combination of selection and optimization algorithms. Such conclusion follows from the sensitivity of LVQ to initialization of codebook vector positions ("codebook" is used in vector quantization as a synonym of reference vector [14]). Selection algorithms usually provide a very good initialization for LVQ optimization algorithm. Although the accuracy is high selection algorithms frequently overestimate the number of codebook required to achieve maximum accuracy. Minimization of the number of codebook vectors increases comprehensibility of the model, allowing to define prototype-based rules (P-rules) based on evaluation of similarity to a small number of prototypes [17]. In [18] context-dependent clustering algorithm has been applied to find appropriate initial position of codebooks vectors. The context of the clustering algorithm was defined to ensure that clusters centroids are placed close to the decision borders. Focus on regions close to the decision border allows for improvement of classification accuracy and reduction of the number of codebook vectors at the same time.

In this paper further improvements of the LVQ optimization procedure along these lines are considered. This is achieved by using training vector weights determined by the clustering context. In the next section a short update on the context-dependent clustering is given, followed by a definition of context and reformulation of LVQ algorithm. Empirical tests on 6 benchmark datasets are presented in section 3, and results are compared to the original LVQ algorithm.

2 Context Dependent Clustering

Context dependent clustering uses context information to guide clustering process. In the simplest case context is described by an external variable that estimates importance of each vector. Conditional Fuzzy C-Means (CFCM) [19] is an extension of the FCM

clustering algorithm, with additional variable \mathbf{y} , defined for every vector \mathbf{x} , that helps to cluster related data. For every vector \mathbf{x}_i the strength of its relation with the external variable y_i is defined by a function $f_i = \mu_A(\mathbf{y}) \in [0, 1]$, where $\mu_A(\mathbf{y})$ is interpreted as a membership function for some fuzzy set A , or simply as a weight that creates clustering condition in context A .

FCM and CFCM are based on minimization of a cost function defined as:

$$J_m(\mathbf{U}, \mathbf{V}) = \sum_{k=1}^{N_c} \sum_{i=1}^n (u_{ki})^m \| \mathbf{x}_i - \mathbf{v}_k \|_A^2 \quad (1)$$

where N_c is the number of clusters centered at \mathbf{v}_k (such centers will be used as reference vectors or the LVQ classifier), n is the number of vectors, $m > 1$ determines clustering fuzziness, and $\mathbf{U} = (u_{ki})$ is a $N_c \times n$ dimensional membership matrix with elements $u_{ki} \in [0, 1]$ defining the membership degree of vector \mathbf{x}_i in cluster \mathbf{v}_k . The matrix \mathbf{U} has to fulfill three conditions:

1° each vector x_i belongs to the k -th cluster to a degree between 0 to 1:

$$u_{ki} \in [0, 1]; k = 1..N_c; i = 1..n \quad (2)$$

2° sum of the membership values of i -th vector x_i in all clusters is equal to f_i

$$\sum_{k=1}^{N_c} u_{ki} = f_i; i = 1..n \quad (3)$$

3° clusters are not empty, and a single cluster does not cover whole space:

$$\sum_{i=1}^n u_{ki} \in (0, n); k = 1..N_c \quad (4)$$

Centers of the clusters are calculated as:

$$\mathbf{v}_k = \sum_{i=1}^n (u_{ki})^m x_i \left/ \sum_{i=1}^n (u_{ki})^m \right.; k = 1..N_c \quad (5)$$

The partition matrix is then calculated using cluster centers, and the cost function (1) is minimized iteratively:

$$u_{ki} = f_i \left/ \sum_{k=1}^C \left(\frac{\| x_i - \mathbf{v}_k \|}{\| x_i - \mathbf{v}_k \|} \right)^{2/(m-1)} \right.; i = 1..n; k = 1..N_c \quad (6)$$

3 Determining the Context

Defining appropriate context should help to find minimum number of LVQ prototypes that provide accurate model of the data. This is an alternative to various editing methods. Support vector machines work quite well because they focus only at the border area, and thus may provide also useful prototypes for LVQ [20]. To encourage selection of

prototypes that lie close to decision border context for CFCM clustering is defined with the help of a coefficient w_k describing distribution of vector positions around \mathbf{x}_k within the same and relatively to other classes. This is defined by the ratio of a within-class scatter to out-of-class scatter:

$$w_i = w(\mathbf{x}_i) = \frac{(n - n_c)}{d \cdot n_c} \frac{\sum_{j, c(\mathbf{x}_i)=c(\mathbf{x}_j)} \|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sum_{j, c(\mathbf{x}_i) \neq c(\mathbf{x}_j)} \|\mathbf{x}_i - \mathbf{x}_j\|^2} \quad (7)$$

Here $c(\mathbf{x}_i)$ denotes class label c_i of the vector \mathbf{x}_i , with n_c vectors in this class, and d is the number of features. For normalized attributes w_i coefficients have values that are close to 0 for vectors inside large homogenous clusters, and close to 1 if the vector \mathbf{x}_i is near the vectors of the opposite classes and far from other vectors from the same class (for example, if it is an outlier). To avoid such situations these weights should be centered around $\mu \in [0.6, 1]$ with a Gaussian membership function, defining the context factor:

$$f_i = f(\mathbf{x}_i) = \exp(-\gamma(w(\mathbf{x}_i) - \mu)^2) \quad (8)$$

with $\gamma \in [0.4, 0.8]$, determined empirically for a wide range of datasets. The μ parameter controls where the prototypes will be placed; for small μ they are closer to the center of the cluster and for larger μ closer to the decision borders. The range in which they are sought is determined by the γ parameter. This function is used to determine the clustering context for CFCM clustering in the weighted LVQ algorithm (WLVQ).

4 Introducing Context in LVQ Algorithm

Classical LVQ1 algorithm is based on the cost function defined as:

$$E(\mathbf{P}) = \frac{1}{2} \sum_{k=1}^{N_c} \sum_{i=1}^n \mathbf{1}(\mathbf{x}_i \in R_k) \mathbf{1}(c(\mathbf{x}_i) = c(\mathbf{p}_k)) \|\mathbf{x}_i - \mathbf{p}_k\|^2 - \frac{1}{2} \sum_{k=1}^{N_c} \sum_{i=1}^n \mathbf{1}(\mathbf{x}_i \in R_k) \mathbf{1}(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)) \|\mathbf{x}_i - \mathbf{p}_k\|^2 \quad (9)$$

where $\mathbf{1}(L)$ identity function returns 1 when condition L is *true*, and 0 otherwise, $c(\mathbf{x})$ returns class label of vector \mathbf{x} , and R_k is the Voronoi area defined for prototype \mathbf{p}_k . This relation can also be interpreted as $\mathbf{x}_i \in R_k$ if $\mathbf{p}_k = \arg \min_{l=1 \dots N_c} (D(\mathbf{x}_i, \mathbf{p}_l))$. This cost function is minimized in respect to position of all prototypes \mathbf{P} using the two update formulas that iteratively change the position of k -th prototype \mathbf{p}_k according to:

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_k + \alpha(j) \mathbf{1}(c(\mathbf{x}_i) = c(\mathbf{p}_k)) (\mathbf{x}_i - \mathbf{p}_k) \\ \mathbf{p}_k &= \mathbf{p}_k - \alpha(j) \mathbf{1}(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)) (\mathbf{x}_i - \mathbf{p}_k) \end{aligned} \quad (10)$$

where $\alpha(j) \in [0, 1]$ is monotonically decreasing step size function and j is the iteration number.

In the original LVQ algorithm only local information is taken into account. Applying the context dependent clustering to LVQ training the $f(\mathbf{x}_i)$ factor is introduced to incorporate some information about overall distribution of vectors from different classes, such that outliers or vectors that appear far from the decision border should have less influence on the value of the cost function. It requires reformulating the cost function, which takes the form:

$$E(\mathbf{P}) = \frac{1}{2} \sum_{k=1}^{N_c} \sum_{i=1}^n \mathbf{1}(\mathbf{x}_i \in R_k) \mathbf{1}(c(\mathbf{x}_i) = c(\mathbf{p}_k)) f(\mathbf{x}_i) \|\mathbf{x}_i - \mathbf{p}_k\|^2 \\ - \frac{1}{2} \sum_{k=1}^{N_c} \sum_{i=1}^n \mathbf{1}(\mathbf{x}_i \in R_k) \mathbf{1}(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)) f(\mathbf{x}_i) \|\mathbf{x}_i - \mathbf{p}_k\|^2 \quad (11)$$

This cost function can be minimized modifying formula (10) such that it takes form:

$$\mathbf{p}_k = \mathbf{p}_k + \alpha(j) f(\mathbf{x}_i) \mathbf{1}(c(\mathbf{x}_i) = c(\mathbf{p}_k)) (\mathbf{x}_i - \mathbf{p}_k) \\ \mathbf{p}_k = \mathbf{p}_k - \alpha(j) f(\mathbf{x}_i) \mathbf{1}(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)) (\mathbf{x}_i - \mathbf{p}_k) \quad (12)$$

where the context factor $f(\mathbf{x}_i)$ describes class-dependent distribution of vectors around \mathbf{x}_i . The $f(\mathbf{x}_i)$ value can also be understood as an instance weight describing the importance of that instance during the training process.

5 Experimental Results

Experiments have been performed on 6 benchmark datasets with real valued attributes obtained from the UCI repository [21]. They include Cleveland Heart Disease, Ionosphere, Pima Indian Diabetes, Sonar, Spambase, and Wisconsin Breast Cancer (WBC).

Table 1. Summary of datasets used in experiments

Title	#Features	#Samples	#Samples per class	Source
Heart	13	297	160 absence / 137 presence	[21]
Ionosphere	34	351	224 / 126	[21]
Diabetes	8	768	500 / 268	[21]
Sonar	60	208	111 metal / 97 rock	[21]
Spambase	57	4601	1813 / 2788	[21]
WBC	9	699	458 / 241	[21]

5.1 Experiment Description

Diagram presenting data flow in computational experiments is presented in Fig (11). In these experiments accuracy of the nearest-prototype classifier with prototypes obtained

from classical LVQ algorithm have been compared with those based on prototypes optimized using the WLVQ algorithm. In both cases prototypes/codebooks were initialized using CFCM clustering algorithm. All processing steps have been embedded in the 10 fold cross-validation procedure. In case of WLVQ algorithm optimal values γ and μ were selected using greedy search algorithm from the 4 values of $\gamma = [0.2, 0.4, 0.6, 0.8]$ and 4 values of $\mu = [0.4, 0.6, 0.8, 1]$.

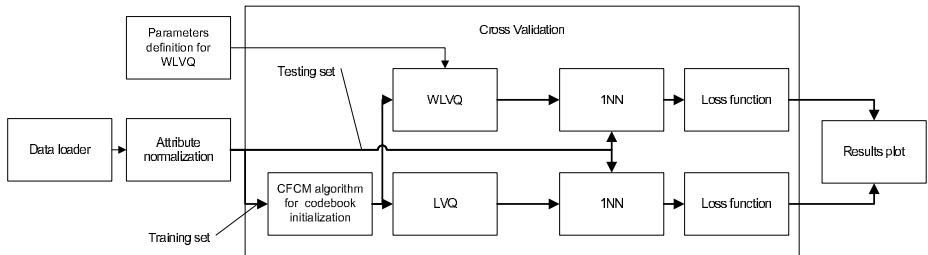


Fig. 1. Data flow diagram

5.2 Results

Results are presented in Fig. 2. Each graph shows the average accuracy and variance as a function of the number of selected prototypes per class. In all plots solid thin line represents accuracy of the reference LVQ algorithm and the bold dotted line represent mean accuracy obtained with WLVQ algorithm.

Results obtained for the *ionosphere* dataset do not show significant improvement in accuracy, but WLVQ has somewhat reduced variance. For *pima diabetes* and *heart disease* datasets average results for the WLVQ algorithm are a bit better than obtained for LVQ, but high variance makes these differences insignificant. The difference can be seen in *sonar* and *spambase* datasets, where especially in the second dataset improvement is significant.

It is worth noting that for *heart disease*, *pima diabetes* and *breast cancer* LVQ finds one prototype per class that is optimal and adding more prototypes is not justified. This leads to a conclusion that linear methods should work well on these datasets, because they are equivalent to prototype classifier with just two prototypes, one per class. This is indeed the case, in all these cases linear solutions do not significantly differ from the best results that have been obtained for these datasets, and the same is true for the LVQ prototypes that may serve as logical rules based on similarity. Such good results may be at least partially attributed to the CFCM codebooks initialization.

For *sonar* 3 prototypes are needed, increasing accuracy by 10%. In this case linear methods are significantly worse than methods based on similarity. In case of *ionosphere* and the *spam* data linear methods completely fail and accuracy is significantly improved by adding more prototypes.

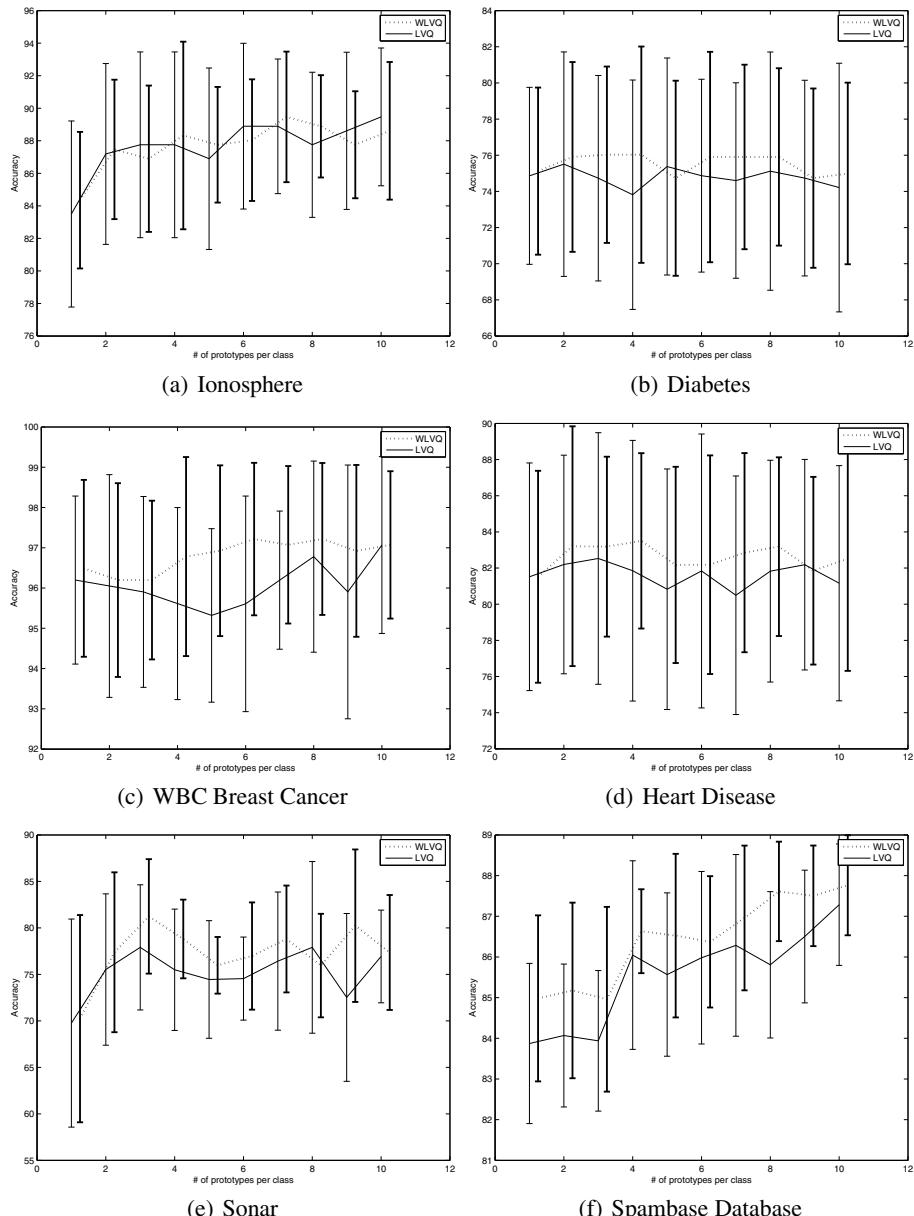


Fig. 2. Comparison of results obtained for LVQ classifier trained with and without instance weighting (both LVQs use CFCM codebooks initialization). Accuracy and standard deviation is presented as a function of the number of codebook vectors per class. Variance of LVQ is shown shifted to the left of the variance of WLVQ.

6 Conclusions

In this paper context dependent clustering has been applied to the LVQ method. We have already shown [22] that the context-dependent clustering may be successfully used to initialize and train neural networks based on localized functions with clearly defined centers, such as the radial basis function networks with Gaussian functions [23][24]. In this case all prototypes generated for network initialization have always been in the vicinity of decision borders. This led to a reduction of the number of prototypes and creation of low-complexity networks. Also application of conditional clustering to represent knowledge contained in the SVM model in a comprehensible way as prototype-based rules (P-rules) led to prototypes that served as support vectors and were placed near the decision border [20]. The SVM hyperplane was used to fit appropriate weights to selected prototypes. Definition of context (Eq. 8) used here is quite simple, and also focused on area around the decision border. Although this definition may be improved in many ways we have now been able to demonstrate that such approach is beneficial for convergence of the LVQ algorithm, leading to improvement of accuracy with reduced variance.

Similarity-based methods provide a powerful framework for knowledge discovery and data mining [23]. They allow to build accurate prediction models in cases where decision borders are strongly non-linear, and to discover knowledge structures in form of similarity to prototypes, or P-rules. These rules are rarely used, although they agree with human intuition based on memory of similar cases seen earlier [25]. Explanatory power of such models may result from optimization of similarity measures optimized for each prototype and finding minimal set of such prototypes that allow for good generalization. P-rule systems make decisions in several ways, but most often the single nearest prototype rule is used. For some datasets a single prototype per class is sufficient, allowing for formulation of a simple classification rule: if the new case \mathbf{x} is more similar to prototype \mathbf{v}_1 rather than to \mathbf{v}_2 it should be assigned to the same class as \mathbf{v}_1 . Such rules may also be converted to fuzzy rules [26].

Selection and optimization of prototypes is an important part of methods that belong to the similarity-based framework. Without such selection the nearest neighbor methods are too slow for real-time applications, they may easily overfit the data, will be hard to use for very large datasets, and lead to data models that are hard to understand. An alternative to good initialization of a few LVQ prototypes is based on editing techniques. Using many prototypes that are pruned at a later stage may be more costly, but direct comparison of such methods with our approach remains to be done. It is clear that training on appropriately pruned data will be especially useful for very large datasets, giving hope to find solutions that are compact, accurate and easy to understand.

Performance of all similarity-based methods, or more generally all methods that calculate distances, including clustering algorithms, is strongly affected by noise that cannot be avoided in high-dimensional problems. To avoid it feature selection or feature aggregation methods should be used as the preliminary step. Simultaneous feature selection, weighting and prototype optimization is of particular importance, although it is still a great challenge. We hope that progress in this direction may be done along similar lines as used in this paper.

Acknowledgment

Authors wants thank to the Polish Ministry of Science and Higher Education to for the financial support of the project no. N516 442138.

References

1. Aha, D.W. (ed.): *Lazy learning*. Kluwer Academic Publishers, Norwell (1997)
2. Duch, W.: Similarity based methods: a general framework for classification, approximation and association. *Control and Cybernetics* 29, 937–968 (2000)
3. Duch, W., Adamczak, R., Diercksen, G.: Classification, association and pattern completion using neural similarity based methods. *Applied Mathematics and Computer Science* 10, 101–120 (2000)
4. Pękalska, E., Paclik, P., Duin, R.: A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research* 2, 175–211 (2001)
5. Schölkopf, B., Smola, A.: *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge (2001)
6. Sonnenburg, S., Raetsch, G., Schaefer, C., Schoelkopf, B.: Large scale multiple kernel learning. *Journal of Machine Learning Research* 7, 1531–1565 (2006)
7. Shakhnarovich, G., Darrell, T., Indyk, P. (eds.): *Nearest-Neighbor Methods in Learning and Vision*. MIT Press, Cambridge (2005)
8. Arya, S., Malamatos, T., Mount, D.: Space-time tradeoffs for approximate nearest neighbor searching. *Journal of the ACM* 57, 1–54 (2010)
9. Wilson, D.: Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Systems, Man and Cybernetics* 2, 408–421 (1972)
10. Wilson, D., Martinez, T.: Reduction techniques for instance-based learning algorithms. *Machine Learning* 38, 257–286 (2000)
11. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery* 6, 153–172 (2002)
12. Bhattacharya, B., Mukherjee, K., Toussaint, G.: Geometric decision rules for instance-based learning problems. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) *PReMI 2005. LNCS*, vol. 3776, pp. 60–69. Springer, Heidelberg (2005)
13. Jankowski, N., Grochowski, M.: Comparison of instance selection algorithms. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) *ICAISC 2004. LNCS (LNAI)*, vol. 3070, pp. 598–603. Springer, Heidelberg (2004)
14. Kohonen, T.: *Self-organizing maps*, 3rd edn. Springer, Heidelberg (2000)
15. Biehl, M., Ghosh, A., Hammer, B.: Dynamics and generalization ability of lvq algorithms. *J. Mach. Learn. Res.* 8, 323–360 (2007)
16. Grochowski, M., Jankowski, N.: Comparison of instance selection algorithms. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) *ICAISC 2004. LNCS (LNAI)*, vol. 3070, pp. 580–585. Springer, Heidelberg (2004)
17. Duch, W., Grudziński, K.: Prototype based rules - new way to understand the data. In: *IEEE International Joint Conference on Neural Networks*, pp. 1858–1863. IEEE Press, Washington (2001)
18. Blachnik, M., Duch, W., Wieczorek, T.: Selection of prototypes rules – context searching via clustering. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Źurada, J.M. (eds.) *ICAISC 2006. LNCS (LNAI)*, vol. 4029, pp. 573–582. Springer, Heidelberg (2006)
19. Pedrycz, W.: *Knowledge-Based Clustering: From Data to Information Granules*. Wiley Interscience, Hoboken (2005)

20. Blachnik, M., Duch, W.: Prototype rules from SVM. Springer Studies in Computational Intelligence, vol. 80, pp. 163–184. Springer, Heidelberg (2008)
21. Asuncion, A., Newman, D.: UCI machine learning repository (2009),
<http://www.ics.uci.edu/~mlearn/MLRepository.html>
22. Blachnik, M., Duch, W.: Building Localized Basis Function Networks Using Context Dependent Clustering. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 953–962. Springer, Heidelberg (2008)
23. Haykin, S.: Neural Networks - A Comprehensive Foundation. Maxwell MacMillian Int., New York (1994)
24. Wilson, D.R., Martinez, T.R.: Heterogeneous radial basis function networks. In: Proceedings of the International Conference on Neural Networks, vol. 2, pp. 1263–1276 (1996)
25. Spivey, M.: The continuity of mind. Oxford University Press, New York (2007)
26. Duch, W., Blachnik, M.: Fuzzy rule-based systems derived from similarity to prototypes. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) ICONIP 2004. LNCS, vol. 3316, pp. 912–917. Springer, Heidelberg (2004)

Multifactor Expectation Maximization for Factor Graphs

Jason T. Rolfe^{1,2} and Matthew Cook²

¹ Computation and Neural Systems,
California Institute of Technology

² Institute of Neuroinformatics,
University of Zurich and ETH Zurich

Abstract. Factor graphs allow large probability distributions to be stored efficiently and facilitate fast computation of marginal probabilities, but the difficulty of training them has limited their use. Given a large set of data points, the training process should yield factors for which the observed data has a high likelihood. We present a factor graph learning algorithm which on each iteration merges adjacent factors, performs expectation maximization on the resulting modified factor graph, and then splits the joined factors using non-negative matrix factorization. We show that this multifactor expectation maximization algorithm converges to the global maximum of the likelihood for difficult learning problems much faster and more reliably than traditional expectation maximization.

1 Introduction

Factor graphs are an attractive formalism for probability distributions over many variables because they permit efficient storage and fast computation of quantities like marginal probabilities [1]. In a factor graph, the full probability distribution over a set of variables is approximated with the normalized product of many functions, each defined over only a small subset of the variables. Unfortunately, their use has been limited at least in part by the difficulty of choosing appropriate parameters on hard problems.

In this paper, we present a method for training factor graphs, based upon the expectation maximization (EM) algorithm [2, 3] and non-negative matrix factorization (NMF) [4], which we have found to give greatly improved performance compared to traditional EM on maximally difficult probability distributions. The main idea is to merge a set of adjacent factors, maximize the likelihood of the observed data over the new joint factor using expectation maximization, and then split the joint factor to return to the original graph topology via non-negative matrix factorization. This is the first application we know of where non-negative matrix factorization methods are used in conjunction with factor graphs. In section 2 we review expectation maximization (technically *generalized* expectation maximization, since only part of the graph is updated at a time) for factor graphs. We derive multifactor expectation maximization as an extension

of traditional EM and non-negative matrix factorization in section 3. Section 4 describes the specific probability distributions, factor graphs, and learning algorithm parameters we used to compare the performance of multifactor EM with single factor EM, and the results of these tests are presented in section 5.

2 Expectation Maximization for Factor Graphs

Consider a set of positive functions f_a with domains $X_a \subset X = Y \cup H$, where Y is the set of observed variables and H is a set of hidden variables not available for direct observation. In a factor graph, these functions are called factors, and are combined to construct a probability distribution over X defined by $p(x) = \frac{1}{Z} \prod_a f_a(x_a)$, where Z is a normalizing factor called the partition function, defined as

$$Z = \sum_X \left(\prod_a f_a(X_a) \right). \quad (1)$$

Z ensures that $\sum_X p(X) = 1$. For simplicity, we consider the case where all variables in X are discrete, although the generalization to continuous variables is straightforward.

Before a factor graph can be used to reason about the world, the functions $f_a(X_a)$ must be chosen. While in some settings, useful systems can be constructed by selecting the f_a manually using expert knowledge [5], in many situations of practical interest, learning must be performed based upon a finite set of observations in a fundamentally unsupervised fashion. If we think of THESE SYSTEMS as simply modeling their environment, this unsupervised learning could take the form of maximizing the likelihood of a set of observed data points $\mathbf{Y} = \{Y_i \in Y\}$. Maximizing the posterior probability of the functions $f_a(X_a)$ given the observed data is equivalent to maximizing the likelihood of the data if the prior probability over the possible $f_a(X_a)$ is flat.

One particularly effective method for maximizing the likelihood of a factor graph over its parameters is the generalized expectation maximization algorithm [6]. The generalized expectation maximization algorithm is an iterative method under which the log likelihood of the observed data ($\log p(\mathbf{Y}|\theta)$, where θ indicates the choice of all $f_a(X_a)$), provably converges to a stationary point, likely a local maximum, even though all operations are performed on a function of the total data (both observed and hidden) [2, 3]. Specifically, we maximize $E[\log p(\mathbf{X}) | \mathbf{Y}, \theta]$ with respect to one factor f_a (for a full derivation, see [7]). The result, also known as iterative proportional fitting (IPF) [8], is

$$f_a(x_a) \leftarrow \frac{f_a(x_a) \cdot \langle p(X_a = x_a | Y, \theta) \rangle}{p(X_a = x_a | \theta)}. \quad (2)$$

3 Multifactor Expectation Maximization

Each iteration of the generalized EM algorithm optimizes an entire factor given the other factors [7, 8], but in practice, learning remains difficult. The large

number of updates required for training with EM is particularly troublesome in loopy graphical models, where it is computationally expensive to compute the marginal probabilities necessary for each EM update [1].

Rather than considering each factor separately, the approach we take here is to group adjacent factors before performing the EM algorithm. Specifically, we replace these adjacent factors with an equivalent new factor over the variables to which they were connected (other than their shared variables), as in Figure 1. We consider here two adjacent factors f_w and f_h defined over pairs variables, $\{i, j\}$ and $\{j, k\}$ respectively, where variable j is only shared by f_w and f_h . This is equivalent to the more general case where f_w and f_h have an arbitrary number of arguments if i and k are understood to be the Cartesian products of the unshared variables of f_w and f_h , respectively. The values of the entries of this new factor are equal to the inner product of the original factors:

$$f_v(i, k) = \sum_j f_w(i, j) \cdot f_h(j, k). \quad (3)$$

Equation (3) can be written in matrix notation as $V = W \cdot H$, where W and H are the matrices containing the entries of the two factors f_w and f_h , with the shared variable j along the columns and rows respectively, and with i and k comprising the other dimension of each matrix (respectively). We use the EM algorithm on this modified factor graph to calculate the desired new values for the joint factor V , which then must be factored to obtain values for the original separate factors. Although the singular value decomposition is known to provide the optimal factorization with respect to the Frobenius norm, it yields negative entries, which are incompatible with the probabilistic interpretation of factor graphs [1]. We thus wish to restrict our factorization to matrices with positive entries, so we use non-negative matrix factorization (NMF) to split the joint factors. So long as the error of the final factorization step is small, this procedure retains all of the performance guarantees of expectation maximization [2, 3]. Furthermore, after each update the shared variable j will be used optimally, given the values of the factors other than f_w and f_h , because the NMF algorithm we use minimizes the error metric defined in equation (4).

A substantial literature exists on non-negative matrix factorization, and many algorithms have been proposed to optimize various error metrics [9, 10]. One of the most parsimonious algorithms is the multiplicative update rule of Lee and Seung [4], which minimizes a generalization of the Kullback-Leibler (KL) divergence:

$$D_{LS}(A||B) = \sum_{ik} \left(A_{ik} \log \frac{A_{ik}}{B_{ik}} - A_{ik} + B_{ik} \right). \quad (4)$$

This generalized KL divergence between a matrix V and its non-negative factorization $W \cdot H$ is locally minimized by the iterated application of the update rules:

$$H_{jk} \leftarrow H_{jk} \frac{\sum_i V_{ik} W_{ij} / (W \cdot H)_{ik}}{\sum_i W_{ij}} \quad (5)$$

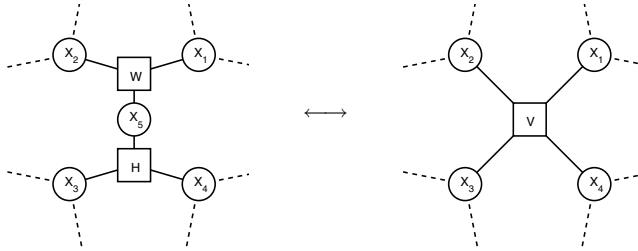


Fig. 1. Schematic joining of two adjacent factors into a multifactor. Squares represent factors and circles represent variables.

$$W_{ij} \leftarrow W_{ij} \frac{\sum_k V_{ik} H_{jk} / (W \cdot H)_{ik}}{\sum_k H_{jk}} \quad (6)$$

Surprisingly, we can also derive equations (5) and (6) directly from the EM update given in equation (2) applied to the subgraph consisting solely of the factors represented by W and H , if we assume that V_{ik} contains the expected marginal probability of the variable combination $\{i, k\}$ given the observed data (i.e., $\langle p(i, k|Y, \theta) \rangle$). Following equation (2), the EM update for f_h can be written as

$$f_h(j, k) \leftarrow f_h(j, k) \cdot \frac{\langle p(j, k|Y, \theta) \rangle}{p(j, k|\theta)} \quad (7)$$

$$\leftarrow f_h(j, k) \cdot \frac{\sum_i \langle p(i, k|Y, \theta) \rangle p(j|i, k)}{\sum_i p(i, j, k|\theta)} \quad (8)$$

$$\leftarrow H_{jk} \cdot \frac{\sum_i V_{ik} W_{ij} H_{jk} / (W \cdot H)_{ik}}{\sum_i W_{ij} H_{jk} / \sum_{ik} (W \cdot H)_{ik}} \quad (9)$$

$$\leftarrow H_{jk} \cdot \frac{\sum_i V_{ik} W_{ij} / (W \cdot H)_{ik}}{\sum_i W_{ij}} \cdot Z \quad (10)$$

where Z is analogous to equation (11). Equation (10) is a scaled version of Lee and Seung's algorithm, and since the probability distribution of a factor graph is not affected when its factors are multiplied by constants, the two are equivalent. The equivalence of the updates for W can be demonstrated via a similar set of manipulations. The ability to rederive equations (5) and (6) from expectation maximization should assuage any concerns regarding the arbitrary nature of the generalized Kullback-Leibler divergence in equation (4).

It is important to note that the update of f_h given in equations (5) and (10), applied to the joint factor, is not equivalent to the EM update using equation (2) applied to the unmodified graph. When performing the EM update on the joint factor V using equation (2), we obtain $V = (W \cdot H)_{ik} \cdot \frac{\langle p(i, k|Y, \theta) \rangle}{p(i, k|\theta)}$, rather than just $\langle p(i, k|Y, \theta) \rangle$, as we assumed for the above derivation. The term $p(i, k|\theta)$ in the denominator depends upon all of the factors, not just f_w and f_h , so the difference

is not local to the updated factors. The application of NMF after performing EM on a modified graph with a joint factor, effectively an EM update nested inside another EM update, is thus not equivalent to the original EM algorithm.

We can extend Lee and Seung's non-negative matrix factorization to cases where more than two factors are connected to the variable subsumed inside the new joint factor. In this case, V is not intrinsically the product of two matrices. Rather, when factors $A_{i_j, x}^j$, $1 \leq j \leq n$ with inputs i_j and x are all connected to variable x , the tensor V_{i_1, i_2, \dots, i_n} representing the new joint factor is defined by $V_{i_1, i_2, \dots, i_n} = \sum_x \prod_j A_{i_j, x}^j$. The update of any single matrix A^j can be performed by rewriting this tensor as an ordinary matrix product $\sum_x A_{i_j, x}^j B_{x, i_1 \times \dots \times i_{j-1} \times i_{j+1} \times \dots \times i_n}^j$, where the second matrix is defined as $B_{x, i_1 \times \dots \times i_{j-1} \times i_{j+1} \times \dots \times i_n}^j = \prod_{k \neq j} A_{i_k, x}^k$. Lee and Seung's update is guaranteed to reduce the KL-like error function when performed on each of W and H separately. Since this error function only depends upon $A^j \cdot B^j = V$, the update of any factor A^j using the B^j as the second factor will indeed reduce an appropriately extended KL-like error function. This algorithm is once again a form of EM.

4 Methods

For measuring the effectiveness of training algorithms, we used a maximally difficult data distribution. We specifically avoided "real life" data sets for which the source of the difficulty of learning is unknown, and which cannot be cleanly scaled to different sizes of factor graph or different statistical complexities.

We investigated the relative performance of single factor and multifactor EM on factor graphs in which each factor involved three variables and each variable was used in two factors. For a factor graph with N input variables Y_1, \dots, Y_N of size M ($Y_i \in \{1, \dots, M\}$), we trained the factor graph to represent a probability distribution where all input combinations satisfying $(\sum_{i=1}^N Y_i) \bmod M = 0$ had equal probability $M^{-(N-1)}$, while all other input combinations had 0 probability.

This is in many ways the most difficult probability distribution on N variables, as only maximal-order statistics are present. If even a single input variable Y_i is unknown, the marginalized joint probability distribution of all the other variables, $\sum_{j=1}^M P(Y|Y_i = j)$, is completely uniform. This means that there are no locally learnable correlations; all correlations are global across all variables. If the hidden variables also take on only M values, then every value of each hidden variable must be used to transmit information, and the entropy of these variables must be maximal. Due to this structure of the probability distribution, all of the factors must be learned simultaneously; it is not possible to correctly learn a subset of the factors first, and then use them to learn the other factors.

To probe the performance of the two algorithms on increasingly large graphs and thus increasingly complex probability distributions, we observed their convergence properties on subgraphs of Figure 2. These are small graphs where the data contains only high-order statistics, and the results in Figure 4(e,f) suggest that the performance on these small graphs can be comparable to that on much

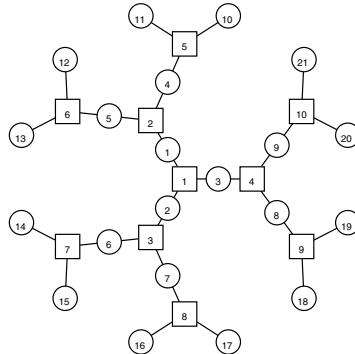


Fig. 2. A factor graph with 10 factors. For learning a distribution on n variables, the subgraph containing factor nodes $1 \dots n - 2$ and variable nodes $1 \dots 2n - 3$ was used, with nodes $1 \dots n - 3$ (those connected to two factors) being the hidden variables and nodes $n - 2 \dots 2n - 3$ (those connected to only one factor) being the observed variables.

larger graphs whose probability distributions have correlations that are mostly local and low-order. We calculated the intrinsic marginal probabilities exactly using belief propagation [1], and estimated the expected marginal probabilities over possible data points by taking the average of the marginals over a batch of trials, with the inputs drawn from the desired probability distribution.

Since the single factors of traditional EM and the joint factors of multifactor EM have 64 and 256 distinct input combinations respectively with four values per variable, we used batches of size 1000 and 4000 (respectively) to estimate the expected marginals given the input distribution. Batch sizes were increased to 2000 and 10000 for single and multifactor EM respectively on runs where variables could take on six values. The batch size does not significantly affect the EM algorithm so long as it is sufficiently large, so we report all convergence times in terms of the number of batches. It is relevant to note that the required size of the batches is proportional to the size of the domain of the (multi-)factors, so multifactor EM will in general require larger batches and thus longer running times for the same number of batches. As will be shown in Figure 4(a,b), multifactor EM for graphs with four values per variable, for which the multifactor batches should be four times larger than the single factor batches, is between 6.7 and 371 times faster than single factor EM, with an average of 100. Figure 4(c,d) suggests that this convergence time ratio grows exponentially quickly as the variable size increases, whereas the difference in batch sizes grows only polynomially. Furthermore, the required batch size can be considerably reduced if the marginal probabilities are approximated by low-pass filtering a sequence of estimates calculated with much smaller batches [11]. Thus, there is strong evidence that the larger batches required for multifactor EM are more than compensated for by the much faster convergence time afforded by the multifactor method.

Most simulations were run for 10^8 iterations or 10^5 batches for single factor EM, and $5 \cdot 10^7$ iterations or $1.25 \cdot 10^4$ batches for multifactor EM. Single factor EM was run for $2 \cdot 10^8$ iterations for graphs with eight factors or six values per variable, and only $5 \cdot 10^7$ iterations for graphs with 3 factors. Multifactor EM was run for up to 10^8 iterations for graphs with eight or nine factors or six values per variable. Multifactor EM was run for fewer iterations only because it converged more rapidly. The finite length of our simulations creates an arbitrary but unavoidable maximum bound on the measured convergence time. Care was taken to allow the simulations to run long enough so that most runs converged well before the end of the simulation, as shown in Figure 3.

In both the single and multifactor EM algorithm, we initialized the factors according to an exponential distribution. Since factor graphs are invariant with respect to constant scaling of their parameters, the exponential distribution is scale-free in the context of factor graphs ($p_a(x \cdot b) = ae^{-a \cdot b \cdot x} \propto p_{ab}(x)$). Our algorithm thus has no parameters other than batch size and number of NMF iterations, both of which need only be chosen to be sufficiently large.

For the multifactor EM algorithm, we iterated equations (5) and (6) 500 times on each update. Empirically, we have found this more than sufficient to reach convergence even on random matrices. The initial values of W and H in the NMF iterations were chosen to be the previous values of the factors. This ensures that the structure of the factors does not needlessly change drastically from update to update and may lead the NMF algorithm to converge more quickly, but it is not necessary.

5 Results

Figure 3 shows the log likelihood averaged over the last $4 \cdot 10^4$ trials versus the number of batches required to reach this likelihood for graphs of various sizes, using both single and multifactor EM. The bands of points in these graphs correspond to the various local maxima found by the algorithms. Although the EM algorithm did occasionally jump from a local maximum to the global maximum, the significant number of trials in which the EM algorithm remained trapped in a local maximum despite the large number of total training cycles suggests that these local maxima can be stable.

Figure 4(a,b) shows the median time to convergence and the probability of convergence to the global maximum as a function of graph size for the single and multifactor EM algorithms. A run was said to have converged to the global maximum if the average log likelihood reached 99% of the actual log probability of the data points in the target distribution ($\log(M^{-(N-1)})$); the time to convergence is the number of batches required to reach this point. Runs that failed to converge were not included in the statistics. Single factor EM did not converge even once in 74 runs on the eight factor graph with up to $2 \cdot 10^5$ batches, so its convergence time is not indicated in Figure 4(a). We report the median time to convergence rather than the average time to convergence because the median reduces the impact of outliers, readily visible in Figure 3, which otherwise obscure the trends in the graphs. Note that the average, represented by an asterisk,

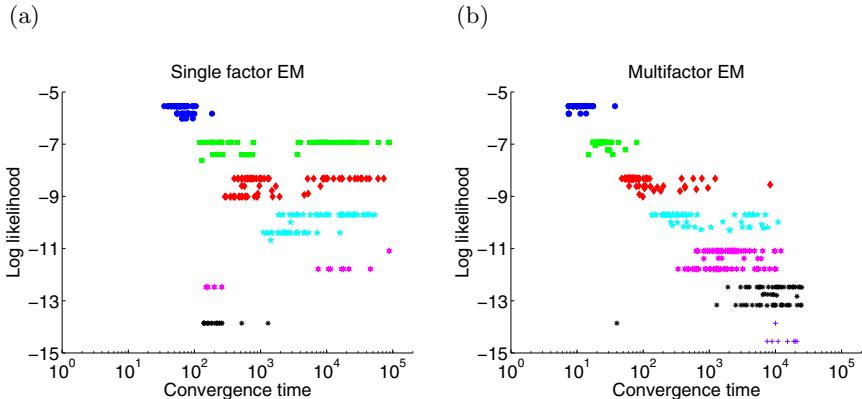


Fig. 3. Number of iterations required to reach the final likelihood versus the log likelihood for (a) single factor and (b) multifactor EM. Runs in which the log likelihood failed to improve at all are not plotted. From top to bottom, blue squares = 3 factors; green circles = 4 factors; red diamonds = 5 factors; cyan five-pointed stars = 6 factors; magenta six-pointed stars = 7 factors; black asterisks = 8 factors; blue +'s = 9 factors.

often falls outside the second and third quartiles of the data, indicated by the whiskers around the median. Multifactor EM converged faster and more reliably for all graph sizes tested. In particular, multifactor EM was 371 times faster on the four-factor graph, and an average of 100 times faster over all of the graph sizes.

Figure 4(c,d) shows the median time to convergence and the probability of convergence to the global maximum for the five factor graph, as a function of the size of the variables. The log of the time required to converge to the global maximum was roughly linear with respect to the size of the variables, suggesting that the time to convergence is exponential in the variable size. The steeper growth of the log convergence time for single factor EM relative to multifactor EM implies that multifactor EM will be exponentially faster as the size of the variables increases. Although the requisite batch size of multifactor EM grows faster than that for single factor EM, these terms are only polynomial in the size of the variables, so multifactor EM should be asymptotically faster. The probability of convergence did not exhibit a consistent trend with respect to variable size, presumably reflecting high-order interactions in the system or effects of the specific graph topology.

As discussed above, the parity-mod- M distribution used in these experiments contains only maximal-order correlations. Most probability distributions of practical interest, such as natural visual and auditory stimuli, have considerable local correlations, and even fourth order correlations are unlikely to appear without lower-order correlations being present as well. To investigate the effect of local structure in the probability distribution, we trained an eight-factor factor graph on the distribution with $(\sum_{i \in A_j} Y_i) \bmod M = 0$ for $A_1 = \{10, 11, 12, 13\}$, $A_2 = \{14, 15, 16, 17\}$, and $A_3 = \{8, 9, 10, 11, 14, 15\}$, in the subset of the graph

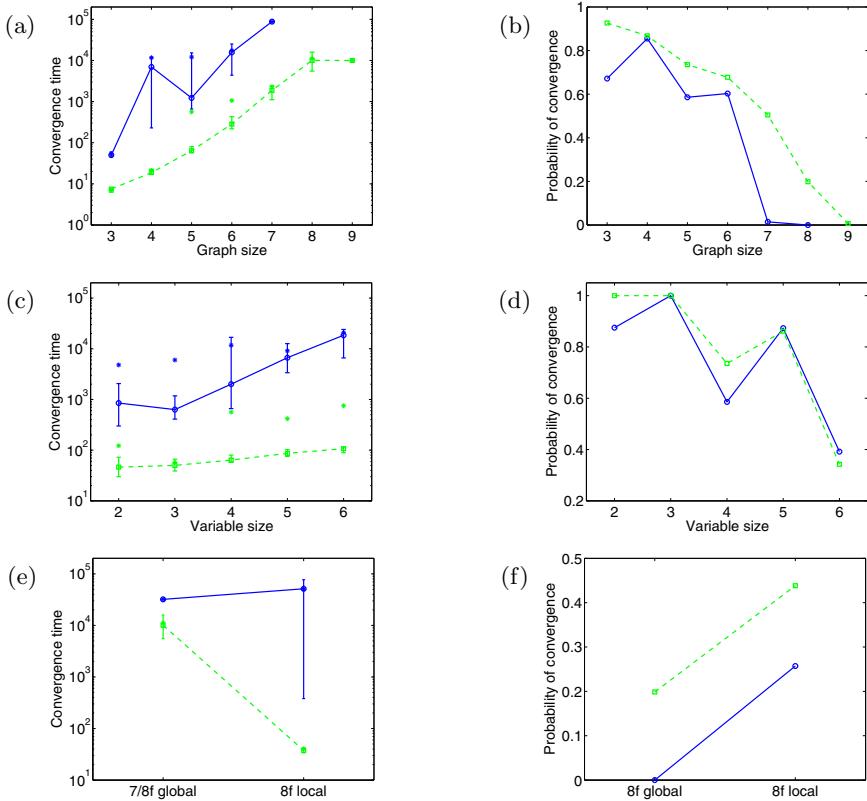


Fig. 4. (a,b) Number of factors in the graph (with variable size four), (c,d) size of variables in the graph (with five factors), and (e,f) global or local correlations in the probability distribution versus (a,c,e) median convergence time and (b,d,f) probability of convergence. See text for details. Blue circles/solid line = single factor EM; green squares/dashed line = multifactor EM. In (a,c,e), the whiskers indicate the second and third quartiles of the data, while the asterisk plots the average.

in Figure 2 with $n = 17$. As can be seen in Figure 4(e,f), this probability distribution was considerably easier to learn than the original parity-mod- M distribution, and the performance advantage of multifactor EM remained. Indeed, single factor EM never converged on the eight-factor global parity distribution; Figure 4(e) shows instead the time for the seven-factor graph as a lower bound.

6 Conclusions

Philosophically, this method for training trees of connected factors is similar to Wainwright's tree reparameterization [12], in that we are taking an iterative,

distributed update algorithm and effectively (although not exactly) rescheduling the updates to reach convergence on a subset of the graph before updating other portions of the graph. The efficacy of multifactor EM can be understood in terms of the properties of non-negative matrix factorization [13]. In choosing the model parameters for two adjacent factors, we are effectively defining equivalence classes of input combinations to the joint factor. These equivalence classes are represented by a common internal message between the two factors. Non-negative matrix factorization can be understood as performing a biclustering operation, and thus intuitively should choose these equivalence classes efficiently.

References

- [1] Kschischang, F., Frey, B., Loeliger, H.: Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory* 47(2), 498–519 (2001)
- [2] Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc B Met.* 39(1), 1–38 (1977)
- [3] Lauritzen, S.: The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis* 19 (1995)
- [4] Lee, D., Seung, H.: Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems* 13, 556–562 (2001)
- [5] Shwe, M., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., Cooper, G.: Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods Inf. Med.* 30(4), 241–255 (1991)
- [6] Frey, B., Jojic, N.: A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 27(9), 1392–1416 (2005)
- [7] Murphy, K.: Dynamic Bayesian networks: Representation, inference, and learning. PhD thesis, University of California, Berkeley (2002)
- [8] Darroch, J., Ratcliff, D.: Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics* 43(5), 1470–1480 (1972)
- [9] Lee, D., Seung, H.: Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755), 788–791 (1999)
- [10] Berry, M., Browne, M., Langville, A., Pauca, V., Plemmons, R.: Algorithms and Applications for Approximate Nonnegative Matrix Factorization. Submitted to *Computational Statistics and Data Analysis* (2006)
- [11] Rolfe, J.: The cortex as a graphical model. Master's thesis, California Institute of Technology (2006)
- [12] Wainwright, M., Jaakkola, T., Willsky, A.: Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. on Information Theory* 49(5), 1120–1146 (2003)
- [13] Brunet, J.P., Tamayo, P., Golub, T., Mesirov, J.: Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences (USA)* 101(12), 4164–4169 (2004)

Weighted Learning Vector Quantization to Cost-Sensitive Learning

Ning Chen¹, Bernardete Ribeiro², Armando Vieira¹,
João Duarte¹, and João Neves³

¹ GECAD, Instituto Superior de Engenharia do Porto, Instituto Politecnico do Porto

² CISUC, Department of Informatics Engineering, University of Coimbra, Portugal

³ ISEG-School of Economics, Technical University of Lisbon, Portugal

{cng,asv,jmmd}@isep.ipp.pt, bribeiro@dei.uc.pt, jcneves@iseg.utl.pt

Abstract. The importance of cost-sensitive learning becomes crucial when the costs of misclassifications are quite different. Many evidences have demonstrated that a cost-sensitive predictive model is more desirable in practical applications than a traditional one without taking the cost into consideration. In this paper, we propose two approaches which incorporate the cost matrix into original learning vector quantization by means of instance weighting. Empirical results show that the proposed algorithms are effective on both binary-class data and multi-class data.

Keywords: classification, neural network, cost-sensitive learning, learning vector quantization.

1 Introduction

Cost-sensitive learning is a particular inductive learning which takes a variety of costs into consideration when making prediction. The prevalently existing misclassification cost plays an important role in decision making and occupies a unique position in cost-sensitive learning. For example, misclassifying a life-threatening disease as healthy is much more serious than false alarm in medical diagnosis. As indicated in [1], there are different categories of misclassification cost, e.g., constant error cost, conditional error cost on individual case, time, other instances or feature. The present-day research concentrates on the former, in which the instances belonging to a class have the same cost. The proposed algorithms in this paper are related to the class-dependent cost formulation. In this paper we focus on the induction of cost-sensitive learning vector quantization (LVQ) based on instance weighting. Two weighted LVQ algorithms are presented, corresponding to the batch training and online training respectively. Firstly, the instances of a class are assigned a weight proportional to corresponding cost. Afterwards, the weights are considered in the training process of a LVQ network. Finally, the network is labeled by a weighted voting principle. A number of data sets including binary-class and multi-class are used in the experiments. The weighted LVQ algorithms are compared with original LVQ in terms of classification error and total cost resulted from the misclassification.

2 Weighted-Instance LVQ Approaches

Learning vector quantization (LVQ) is a neural network for supervised learning. Since the original formulation of LVQ, a lot of variants have been implemented to improve convergence and diminish errors, by means of scaling weight factors of input dimensions with respect to relevance (generalized LVQ) [2], replacing nearest neighbor distance with harmonic average distance (initialization insensitive LVQ) [3], selecting attributes and training data in LVQ scheme [4]. Some attempts have been conducted on the extension of LVQ for cost-sensitive learning [5][6]. However, they concentrate on the binary-class problem. Few were reported regarding cost-sensitive learning on multi-class data using LVQ. In this section, we extend the original LVQ algorithm by integrating instance weights into the learning and labeling of the network in a simple way, thus induce a cost-sensitive LVQ classifier.

A widely applied method is introducing some weights into the underlying learning algorithms [7]. The main issue is how to set appropriate values of the weights. Normally, the cost of misclassification can be represented in a matrix. In this study, we simplify the cost matrix to a cost vector, where S_i means the cost of misclassifying a class i instance to others. The conversion is intuition by summing the values of the cost matrix by rows followed by normalization so that the minimal cost is one. Let N be the number of instances in the training data, N_j the number of class j instances. The weights of instances are calculated from the cost vector.

$$w(x_i) = S_i \frac{N}{\sum_j S_j N_j}$$

Based on LVQ1 and batch LVQ [8], we propose two weighted learning vector quantization approaches, namely wLVQo (online weighted LVQ) and wLVQb (batch weighted LVQ) are presented. In the former, the prototypes are updated as: $m_p = m_p \pm \alpha w(x_i)(x_i - m_p)$ (+ if x_i have the same class with m_p , – otherwise) in which α denotes the learning rate. In the latter, the instance weights are incorporated into the updating of prototypes so that instances with higher weights in the Voronoi set V_p have more influence on neuron m_p .

$$s_{ip} = \begin{cases} 1 & \text{if } Class(m_p) = Class(x_i) \\ -1 & \text{otherwise} \end{cases} \quad m_p = \frac{\sum_{x_i \in V_p} w(x_i) s_{ip} x_i}{\sum_{x_i \in V_p} w(x_i) s_{ip}}$$

The weighted voting principle is an extension of majority voting. The training instances are projected to the trained map via the nearest neighbor principle, then the number of the instances are aggregated for each different class. Let w_j be the instance weight assigned to class j , N_j the number of class j instances in the Voronoi set of the neuron i . The voting of a class is calculated as the occurrence multiplied by the weights, consequently, the label with the highest vote is assigned to the neuron.

$$Class(m_i) = \operatorname{argmax}_j N_j w_j$$

3 Empirical Analysis

In the experiments, the data sets are taken from UCI [9] except *Diane*. Since the weighted LVQ algorithms are presented for numeric data, the categorical features contained in *horse* and *credit* are converted to several binary ones by means of one categorical value corresponding to one binary feature in the new data set. *Diane* [5] is real-world data set containing information on a wide set of financial ratios spanning over a period of several years. The data used in this study is a balanced subset composed of 600 distressed companies and 600 healthy ones. We use the overall error rate (*Err*), the error rate of the highest costly class (*Err_{HC}*) and the total cost (*TC*) on all instances to evaluate the performance.

The experiments are performed in the following strategy. Firstly, the cost matrix is generated so that the diagonal values are 0 and others are random number taken from [1,10]. The data set is divided randomly into ten folds for cross-validation. In each trial one fold is used as test data, and the remaining is used as the training data. For each generated training data set, the weighted LVQ approaches are applied. Afterwards, the test data is input to the resultant map, and the class is predicted via the nearest neighbor principle. After the cross-validation is finished, the contingency matrix is obtained by summarizing the real class and predicted class for the entire data. Then the criteria described above are calculated from the contingency matrix. The process is repeated 10 times with randomly generated cost matrix, and the average results are calculated.

Table 1. Performance comparison (at the * 1%, ** 5%,*** 10% significance level)

Data set	# ins.	# att.	# cla.	<i>Err_{HC}</i>		<i>Err</i>		<i>TC</i>	
				wLVQb	wLVQo	wLVQb	wLVQo	wLVQb	wLVQo
binary-class									
Diane	1200	30	2	0.56*	0.68**	1.20	1.52	0.80**	1.00
transfusion	748	5	2	0.70*	0.71*	1.06	1.11	0.91*	0.96
breast	699	10	2	0.94	0.91	1.00	1.08	0.95	1.02
haberman	306	3	2	0.64**	0.60**	1.09	1.19	0.91*	0.95
ionosphere	351	34	2	0.91***	0.92	1.01	1.50	0.96	1.22*
diabetes	768	8	2	0.57*	0.60**	1.18	1.29	0.95	1.05
horse	368	23	2	0.89**	1.01	1.04	1.32	0.96	1.17**
credit	690	15	2	0.67*	0.71*	1.04	0.97	1.05	1.02
mean				0.74	0.77	1.08	1.25	0.94	1.05
multi-class									
glass	241	10	7	0.67*	0.71*	1.04	0.97	1.05	1.02
segmentation	210	19	7	0.71	0.63	1.12	1.00	1.02	0.95
iris	150	4	3	0.88	0.81	1.02	1.08	0.95	0.95
vehicle	0.7	94	18	48**	0.75*	1.10	0.97	1.02	0.93
yeast	1484	8	10	0.88*	0.96	1.01	1.03	1.00	1.02
lung-cancer	32	56	3	0.90**	0.71**	0.96	1.12	0.95	0.81**
ecoli	336	7	7	0.92	0.89	1.04	1.33	1.00	1.23
mean				0.8	0.78	1.04	1.07	0.99	0.99

The ratios of wLVQb/LVQb and wLVQo/LVQb (LVQb denotes original batch LVQ) in terms of high cost error rate, overall error rate, and total cost are shown in Table I. A value smaller than 1 means an improvement and the significance is validated by t-test. The means of the ratios are given for the 8 binary-class data sets and 7 multi-class data sets respectively. The two weighted LVQ approaches are capable to lower the high cost error rate, with a slight degradation on the overall error rate. In terms of high cost error, almost all data sets except *horse* have improvement on this ratio (Particularly, wLVQb achieves significantly better results on 7 out of 8 binary-class data sets, and 4 out of 7 multi-class data sets. Regarding wLVQo, the results are 5 and 3 respectively). The mean reduction is 26% for wLVQb and 23% for wLVQo, indicating the effectiveness of weighted LVQ approaches. The decrease on total cost is not significant implying the fact that the proposed approaches aim to improve the classification capability on costly class rather than the total cost directly. The performance on binary-class data is superior over that on multi-class data. The relative poor performance is probably due to the information loss during the conversion from cost matrix to class weights. This result is consistent with what was reported in [7]. The two weighted LVQ approaches perform comparably, in which wLVQb achieves better results on binary-class, while wLVQo achieves better results on multi-class.

4 Conclusions

The weighted LVQ approaches use instance weighting to induce a cost-sensitive LVQ classifier. With well-defined weight assignment, the cost matrix can be incorporated in the training and labeling of the LVQ model. Results on a number of real-world data demonstrate the presented weighted LVQ approaches are effective on both binary-class and multi-class data. Future work will consider the adaption of proposed strategy on advanced LVQ variants and better incorporating the cost matrix in the representation of instance weight. The comparative study with other cost-sensitive learning methods is needed to validate the effectiveness of the proposed approaches.

Acknowledgements

This work was supported by project C2007-FCT/442/2006-GECAD/ISEP (Knowledge Based, Cognitive and Learning Systems) and PTDC/GES/70168/2006.

References

1. Turney, P.D.: Types of Cost in Inductive Concept Learning. In: Workshop on Cost-Sensitive Learning at 7th International Conference on Machine Learning, California, pp. 15–21 (2000)
2. Hammer, B., Villmann, T.: Generalized Relevance Learning Vector Quantization. Neural Networks 15, 1059–1068 (2002)

3. Qin, A., Suganthan, P.: Initialization Insensitive LVQ Algorithm based on Cost-Function Adaptation. *Pattern Recognition* 38(5), 773–776 (2005)
4. Pedreira, C.E.: Learning Vector Quantization with Training Data Selection. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 28(1), 157–162 (2006)
5. Chen, N., Vieira, A., Duarte, J.: Cost-sensitive LVQ for Bankruptcy Prediction: An Empirical Study. In: Li, W., Zhou, J. (eds.) 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, pp. 115–119 (2009)
6. Chen, N., Vieira, A., Duarte, J., Ribeiro, B., Neves, J.C.: Cost-sensitive Learning Vector Quantization for Financial Distress Prediction. In: Lopes, L.S., Lau, N., Mariano, P., Rocha, L.M., et al. (eds.) EPIA 2009. LNCS (LNAI), vol. 5816, pp. 374–385. Springer, Heidelberg (2009)
7. Ting, K.M.: An Instance-Weighting Method to Induce Cost-Sensitive Trees. *IEEE Transactions on Knowledge and Data Engineering* 14(3), 659–665 (2002)
8. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Heidelberg (2001)
9. UCI Machine Learning Repository,

<http://www.ics.uci.edu/~mlearn/MLRepository.html>

Solution Space of Perceptron

Jau-Chi Huang and Cheng-Yuan Liou^{*}

Department of Computer Science and Information Engineering
National Taiwan University
Taiwan, Republic of China
Supported by National Science Council
cyliou@csie.ntu.edu.tw

Abstract. Any point in the solution space of a perceptron can classify the training data correctly. Two kinds of the solution space, one in the weight space and the other in the input space, have been devised. This work illustrated the correspondence between these two spaces.

1 Introduction

Consider a neuron [1], [2] with its weights $[w_1, w_2, w_3]$ and two inputs $[x_1, x_2]$, plus a fixed input 1 for the threshold w_3 . In Fig. 1(a), L designates a decision line given by weights $[w_1, w_2, w_3]$. This line, $L : w_1x_1 + w_2x_2 + w_3 = 0$, can be represented by a perpendicular point at the location (a_1, a_2) , where $a_1 = \frac{-w_1w_3}{w_1^2+w_2^2}$ and $a_2 = \frac{-w_2w_3}{w_1^2+w_2^2}$. Accordingly, each point (a_1, a_2) represents two decision lines, they are $S \left[\frac{a_1}{a_1^2+a_2^2}x_1 + \frac{a_2}{a_1^2+a_2^2}x_2 - 1 \right] = 0$, $S = \pm 1$. In the n -dimensional input space, a decision hyperplane given by weights $[w_1, w_2, \dots, w_{n+1}]$ can be represented by a point (a_1, a_2, \dots, a_n) on the input space, where

$$a_k = \frac{-w_k w_{n+1}}{w_1^2 + w_2^2 + \dots + w_n^2}. \quad (1)$$

Each point represents two decision hyperplanes,

$$S \left[\frac{a_1}{a_1^2 + \dots + a_n^2}x_1 + \dots + \frac{a_n}{a_1^2 + \dots + a_n^2}x_n - 1 \right] = 0, S = \pm 1. \quad (2)$$

Through this representation method, we can map the whole weight space to the points in the input space and study the solution space. One can draw the global error surfaces and interpret various learning behaviors [3], [4].

2 The Solution Space

Consider a set of n -dimensional patterns $\tilde{P} = \{\tilde{p}_1 \dots \tilde{p}_N\}$. These patterns belong to two classes, class A and class B . We add one extra dimension for

* Corresponding author.

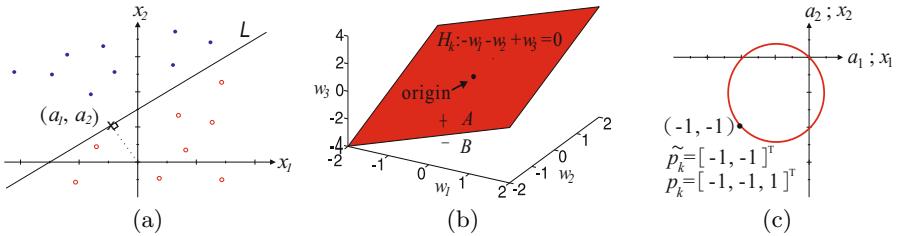


Fig. 1. (a) The decision line L can be represented by a perpendicular point (a_1, a_2) . (b) The hyperplane, $H_k: -w_1 - w_2 + w_3 = 0$, for the pattern: $p_k = [-1, -1, 1]^T$. (c) The mapped circle in the input space, $C_k: (a_1 + 0.5)^2 + (a_2 + 0.5)^2 = 0.5$, which passes the origin and the pattern $\tilde{p}_k = [-1, -1]^T$.

each pattern as the fixed input 1 that is used for the threshold, and get a set $n + 1$ -dimensional patterns, $P = \{p_1 \dots p_N\}$ where $p_k = [p_{k,1}, p_{k,2}, \dots, p_{k,n}, p_{k,n+1}]^T = [\tilde{p}_{k,1}, \tilde{p}_{k,2}, \dots, \tilde{p}_{k,n}, 1]^T$. When there exists a neuron with its weight vector, $w = [w_1, w_2, \dots, w_{n+1}]^T$, which can classify the patterns successfully, then $w^T p_i > 0$, if $p_i \in A$; $w^T p_i < 0$, if $p_i \in B$. The case, $w^T p_i = 0$, is omitted to simplify the expression in this paper.

In the weight space, the hyperplane for the pattern p_k , $\{H_k: w^T p_k = 0\}$, passes through the origin. Any vector w which belongs to the positive half-space of H_k will classify p_k as class A , because of $w^T p_k > 0$. Any vector w which belongs to the negative half-space of H_k will classify p_k as class B , because of $w^T p_k < 0$, see Fig. 1(b). While applying the representation method 3, the hyperplane H_k in the weight space will map to a circle C_k in the input space which passes the origin and the point \tilde{p}_k , see Fig. 1(c). The proof is as follows.

Rewriting the hyperplane equation, H_k , we get $w_{n+1} = -(w_1 p_{k,1} + w_2 p_{k,2} + \dots + w_n p_{k,n})$. Using 1, obtain

$$(a_1^2 + \dots + a_n^2) - (a_1 p_{k,1} + \dots + a_n p_{k,n}) = \\ \frac{w_{n+1}^2 (w_1^2 + w_2^2 + \dots + w_n^2)}{(w_1^2 + w_2^2 + \dots + w_n^2)^2} + \frac{w_{n+1} (w_1 p_{k,1} + w_2 p_{k,2} + \dots + w_n p_{k,n})}{w_1^2 + w_2^2 + \dots + w_n^2} = 0.$$

Rearranging terms, we get the circle equation

$$C_k : (a_1 - \frac{p_{k,1}}{2})^2 + (a_2 - \frac{p_{k,2}}{2})^2 + \dots + (a_n - \frac{p_{k,n}}{2})^2 = \\ (a_1^2 + \dots + a_n^2) - (a_1 p_{k,1} + \dots + a_n p_{k,n}) + \frac{p_{k,1}^2 + \dots + p_{k,n}^2}{4} = \frac{p_{k,1}^2 + \dots + p_{k,n}^2}{4}.$$

The center of the circle is $(\frac{p_{k,1}}{2}, \dots, \frac{p_{k,n}}{2})$ and its radius is $\sqrt{\frac{p_{k,1}^2 + \dots + p_{k,n}^2}{4}}$.

On the other hand, the equation for any point inside the circle is

$$(a_1 - \frac{p_{k,1}}{2})^2 + (a_2 - \frac{p_{k,2}}{2})^2 + \dots + (a_n - \frac{p_{k,n}}{2})^2 - \frac{p_{k,1}^2 + \dots + p_{k,n}^2}{4} < 0.$$

Rewrite the inside equation, $(a_1^2 + \dots + a_n^2) - (a_1 p_{k,1} + \dots + a_n p_{k,n}) < 0$. This equation in terms of weights is

$$\frac{w_{n+1}[w_{n+1} + (w_1 p_{k,1} + w_2 p_{k,2} + \dots + w_n p_{k,n})]}{w_1^2 + w_2^2 + \dots + w_n^2} < 0.$$

Since the denominator is positive, the class information is in the numerator. Note that when $S = +1$ the threshold is a negative value, $w_{n+1} < 0$. When $S = -1$, the threshold is a positive value, $w_{n+1} > 0$. With some manipulations, we obtain two formulas for the classification,

$$\begin{cases} \text{If } S = +1, \text{ then } w_1 p_{k,1} + w_2 p_{k,2} + \dots + w_n p_{k,n} + w_{n+1} > 0; \\ \text{If } S = -1, \text{ then } w_1 p_{k,1} + w_2 p_{k,2} + \dots + w_n p_{k,n} + w_{n+1} < 0. \end{cases}$$

Therefore, under the condition of $S = +1$, the solution space is the intersection of the inside regions of C_i where $\tilde{p}_i \in A$ and the outside regions of C_j where $\tilde{p}_j \in B$. Under the condition of $S = -1$, the solution space in the input space is the intersection of the inside regions of C_j where $\tilde{p}_j \in B$ and the outside regions of C_i where $\tilde{p}_i \in A$. The union of those solution spaces under $S = +1$ and $S = -1$ will be the whole solution space of patterns $P = \{\tilde{p}_1 \dots \tilde{p}_N\}$ in the input space, Fig. 2(a). When patterns are linearly non-separable, the intersection will not exist, Fig. 2(b).

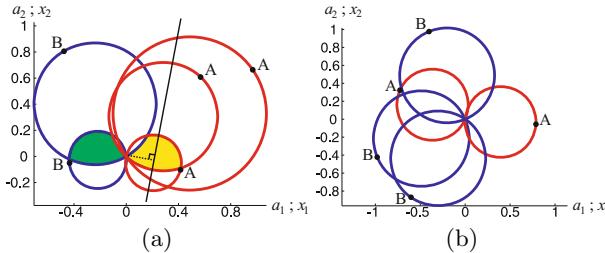


Fig. 2. (a) The solution space exists. The decision line from any perpendicular point in the solution space can separate the patterns successfully. The yellow area is for $S = +1$, and the green area is for $S = -1$. (b) The solution space doesn't exist. The patterns are linearly non-separable.

3 The Correspondence

In the weight space, a *solution cone* was introduced [5]. When we redefine the set P' with elements derived from the elements of patterns P as $p'_i = p_i$, if $p_i \in A$; $p'_i = -p_i$, if $p_i \in B$, then $w^T p'_i > 0$ for all i . In the weight space, for a pattern p'_k , the hyperplane $H'_k : w^T p'_k = 0$ will pass through the origin and any vector w in the positive half-space of H'_k can classify p'_k correctly. Therefore, the intersection of all positive half-spaces of hyperplanes $\{H'_1, \dots, H'_N\}$ will be the solution space of P' . When $n = 2$, the solution space in 3D weight space will have a cone shape with its vertex at the origin and the facets of the cone are

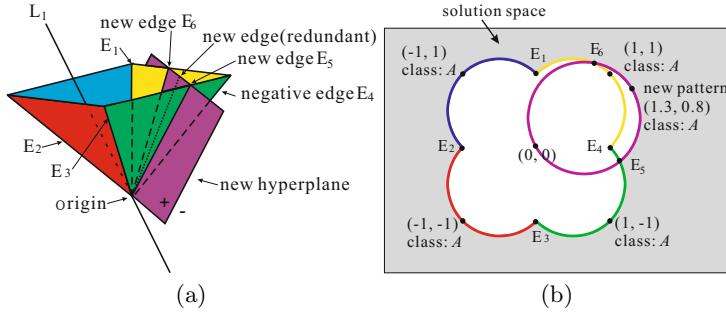


Fig. 3. Consider four patterns $\{\tilde{p}_1 = [-1, 1]^T, \tilde{p}_2 = [1, 1]^T, \tilde{p}_3 = [-1, -1]^T, \tilde{p}_4 = [1, -1]^T\}$ and the added pattern $\tilde{p}_5 = [1.3, 0.8]^T$. They all belong to class A . (a) The solution cone was cut when the new plane, $\{1.3w_1 + 0.8w_2 + w_3 = 0\}$, is added. (b) The border of the solution space changed when this new pattern is included.

the fan sectors of the hyperplanes which belong to the subset of $\{H'_1, \dots, H'_N\}$. When patterns are linearly non-separable, the intersection cone will not exist.

In the paper [5], the authors used edges to define the solution cone. By adding new edges and removing negative edges when each pattern (hyperplane) is presented, they obtain the final solution cone which was defined by the remained edges after all patterns were presented, see Fig. 3(a). There exist a correspondence between the weight space and the input space. When the dimension is $n = 2$, a line in the 3D weight space is a point in the input space. For example, the line $\{L_1 : w_1 = -0.5k, w_2 = 0.5k, w_3 = -k\}$ in Fig. 3(a) will map to the point $(-1, 1)$ in Fig. 3(b). Moreover, a facet of the solution cone in the 3D weight space is an arc in the 2D input space. One edge of the cone in the 3D weight space is an intersection point of two arcs in the 2D input space.

This paper presented a technique to visualize the global structure of the solution space for perceptron. To our knowledge, this solution space is the only one that can be used to explain various training behaviors of perceptron [4].

References

1. McCulloch, W.S., Pitts, W.: A Logical Calculus on the Ideas Immanent in the Nervous Activity. *Bull. Math. Biophys.* 5, 115–133 (1943)
2. Minsky, M., Papert, S.: *Perceptrons*. MIT Press, Cambridge (1969)
3. Liou, C.-Y., Chen, H.-T.: Self-Relaxation for Multilayer Perceptron. In: The Third Asian Fuzzy Systems AFSS 1998, Masan, Korea, June 18–21, pp. 113–117 (1998)
4. Liou, C.-Y., Huang, J.-C., Kuo, Y.-T.: Geometrical Perspective on Learning Behavior. *Journal of Information Science and Engineering* 21, 721–732 (2005)
5. Diamantaras, K.I., Strintzis, M.G.: Neural Classifiers Using One-Time Updating. *IEEE Trans. Neural Networks* 9(3), 436–447 (1998)

Natural Language Processing Neural Network for Recall and Inference

Tsukasa Sagara and Masafumi Hagiwara

Department of Information and Computer Science, Keio University,
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522 Japan
{sagara,hagiwara}@soft.ics.keio.ac.jp

Abstract. In this paper, we propose a novel neural network which can learn knowledge from natural language documents and can perform recall and inference. The proposed network has a sentence layer, a knowledge layer, ten kinds of deep case layers and a dictionary layer. In the network learning step, connections are updated based on Hebb's learning rule. The proposed network can handle a complicated sentence by incorporating the deep case layers and get unlearned knowledge from the dictionary layer. In the dictionary layer, *Goi-Taikei*, containing 400,000 words dictionary, is employed. Two kinds of experiments were carried out by using *goo* encyclopedia and Wikipedia as knowledge sources. Superior performance of the proposed neural network has been confirmed.

Keywords: Neural Network, Natural Language Processing, Inference.

1 Introduction

Natural Language Processing using Neural Networks has been studied [1, 2, 3, 4, 5, 6, 7]. For example, [5] can perform inference based on cognitive science, however, it does not obtain knowledge from natural language documents. [6, 7] can get knowledge from natural language documents and perform simple inference. However, the natural language sentences used in them consist of about a few word, and they cannot handle the complicated sentences. In addition, their inference's precision is not sufficient. This paper proposes a novel neural network which can learn knowledge from natural language documents and can perform recall and inference.

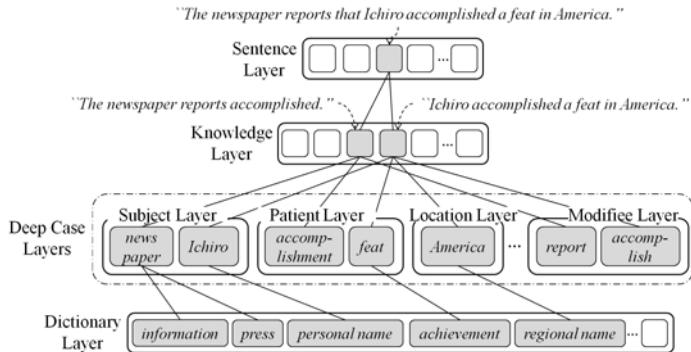
Following this introduction, the proposed network is explained in Sec. 2. Then, experimental results are shown in Sec. 3. Sec. 4 concludes the paper.

2 Proposed Natural Language Processing Neural Network (NLPNN)

The proposed NLPNN consists of three steps: preprocessing step for input sentences; network learning step; question answering processing step.

Table 1. Result of preprocessing for the sentence [S1]

Sentence	Knowledge Unit	Word	Deep Case
The newspaper	The newspaper	newspaper	Subject
reports	reports	report	Modifiee
that	an accomplishment	accomplishment	Patient
Ichiro	Ichiro	Ichiro	Subject
accomplished	accomplished	accomplish	Modifiee
a feat	a feat	feat	Patient
in America	in America	America	Location

**Fig. 1.** Network after neuron energy is propagated

In preprocessing step, input sentences are divided into knowledge units, and the deep cases included in the knowledge are estimated. The techniques to divide the sentences into knowledge units and to estimate the deep case are based on [8,9,10,11]. Table 1 shows an example of the preprocessing for the sentence [S1].

[S1] The newspaper reports that Ichiro accomplished a feat in America.

In the network learning step, the neuron energy is propagated, and the connections are updated. Figure 1 shows the network after the neuron energy is propagated when the sentence [S1] is input. The sentence layer's neurons obtained from the sentence are given the initial energy E_{init} . The connections between layers are given the initial connection weight w_{init} . The energy propagation starts from the sentence layer's neurons and spreads through the connections in sequence. $E_j^{(t)}$ is updated based on $E_i^{(t)}$ and w_{ji} . The neuron i fires if $E_i^{(t)}$ is larger than the fire threshold. w_{ji} is updated based on Hebb's learning rule when both neurons i and j fire. When the energy propagation is completed, the energy forgetting is carried out.

In question answering step, the proposed network performs recall and inference. When a natural language question sentence is input, preprocessing is carried out in the same way as the input sentences.

Table 2. Some examples of question sentences and the output answers

	Question sentence	System's Answer	Correct Answer
(a)	Do fishes breathe with underwater oxygen through gills?	YES	YES
	Do birds form a nest ashore and lay an egg with a hard husk?	YES	YES
(b)	Are dogs fin?	NO	NO
	Do fishes breathe with underwater oxygen through lungs?	YES	NO
(c)	Do Killifishes live in the water?	YES	YES
	Do crows have a thin skin?	YES	YES
(d)	Are dogs covered with fur, and do they retain body heat?	YES	YES
	Do crows have scaled feet and a beak covered with a keratinous sheath?	NO	YES

Table 3. Averaged recall and inference average rate [%]

	Question(a)	Question(b)	Question(c)	Question(d)
Ref. [7]	86.3	No Data	80.4	NoData
Proposed	99.7	92.0	100.0	52.9

In the judgement of recall phase, the energy propagation starts from the neurons of the subject case and the modifiee case obtained from the question sentence. The neurons are given the initial energy E_{init} . The energy propagation spreads through the connections in sequence. In this judgment of recall phase, only learned knowledge is used for the judgement. Therefore, activation is not propagated to the dictionary layer.

In the judgement of inference phase, there is the energy propagation to the dictionary layer. It is very difficult to get correct answer using only information contained in the input sentences. To cope with this problem, the proposed network employes the dictionary layer. It can offer huge amount of knowledge stored in the *Goi-Taikei* [12] dictionary.

3 Evaluation Experiments

The evaluations of recall and inference were carried out to evaluate the proposed network by comparing with the conventional network [7]. In each experiment, *goo* dictionary [13] and Wikipedia encyclopedia [14] were used as knowledge source. They include 4,522 words and 7,497 characters.

In recall, (a) 286 learned sentences were used as questions (Answers are “YES”) and (b) 50 pseudo-learned sentences, slightly changed so that their answers were “NO”, were used as questions. In inference, (c) Relatively simple 15 sentences including two knowledge units and (d) Relatively complex 15 sentences including more than two knowledge units were used as questions. Table 2 shows some examples of question sentences used in the experiment and the output answers. Table 3 summarizes the averaged recall and inference rate.

4 Conclusions

A novel neural network has been proposed in this paper. It can learn knowledge from natural language documents and can perform recall and inference.

The proposed network has two distinctive features. One is usage of ten kinds of deep cases to treat complicated sentences. The other is usage of a dictionary layer. The dictionary layer uses knowledge obtained from *Goi-Taikei* which contains 400,000 words dictionary.

Two kinds of experiments were carried out by using *goo* encyclopedia and Wikipedia as knowledge source. Superior performance of the proposed neural network has been confirmed.

References

1. Tamura, A., Anzai, Y.: Natural language processing system based on connectionist models. IPSJ 28(2), 202–210 (1987)
2. McClelland, J., Rumelhart, D.: Explorations in parallel distributed processing: a handbook of models, programs, and exercises. MIT Press, Cambridge (1988)
3. Cangelosi, A., Domenico, P.: The processing of verbs and nouns in neural networks. Brain and Language 89, 401–408 (2004)
4. Laukaitis, R., Laukaitis, A.: Natural Language Processing and the Conceptual Model Self-organizing Map. In: Kedad, Z., Lammari, N., Métais, E., Meziane, F., Rezgui, Y. (eds.) NLDB 2007. LNCS, vol. 4592, pp. 193–203. Springer, Heidelberg (2007)
5. Hummel, J., Holyoak, K.: A symbolic-connectionist theory of relational inference and generalization. Psychological Review 110, 220–264 (2003)
6. Sakakibara, K., Hagiwara, M.: A proposal of 3-dimensional self-organizing memory and its application to knowledge extraction from natural language. Transactions of the Japanese Society for Artificial Intelligence: AI 21, 73–80 (2006)
7. Saito, M., Hagiwara, M.: Natural language processing neural network for analogical inference. IEICE Technical Report. NC. 108(480) , 1–6 (2009)
8. Kudo, T., Matsumoto, Y.: Japanese dependency analysis using cascaded chunking. In: CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops), pp. 63–69 (2002)
9. Watanabe, T., Ohta, M., Ohta, K., Ishikawa, H.: A page fusion method using case grammar. DBSJ 2004(72), 653–660 (2004)
10. Watanabe, T., Ohno, S., Ohta, M., Katayama, K., Ishikawa, H.: A Distinction Emphasis Multi-document Fusion Technique. In: DEWS 2005 (2005)
11. Oishi, T., Matsumoto, Y.: Lexical knowledge acquisition for japanese verbs based on surface case pattern analysis. IPSJ 36, 2597–2610 (1995)
12. Shirai, S., Ooyama, Y., Ikebara, S., Miyazaki, M., Yokoo, A.: Introduction to Goi-Taikei: A Japanese Lexicon. IPSJ SIG Notes 98, 47–52 (1998)
13. NTT Resonant Inc.: goo dictionary, <http://dictionary.goo.ne.jp/>
14. Wikipedians: Wikipedia, <http://ja.wikipedia.org/wiki/>

Nominally Conditioned Linear Regression

Yusuke Tanahashi¹, Ryohei Nakano², and Kazumi Saito³

¹ Nagoya Institute of Technology, Gokiso, Showa, Nagoya, 466-8555 Japan
tanahasi@ics.nitech.ac.jp

² Chubu University, 1200 Matsumoto, Kasugai 487-8501 Japan
nakano@cs.chubu.ac.jp

³ University of Shizuoka, Yada, Suruga, Shizuoka, 422-8526 Japan
k-saito@u-shizuoka-ken.ac.jp

Abstract. This paper proposes a method for finding a set of regression rules to fit data containing nominal variables as well as numerical ones. Here a regression rule is a linear regression function accompanied with the corresponding nominal condition. A set of such rules can be learned by a four-layer perceptron. A couple of model parameters are selected based on the BIC. In our experiments using 11 real data sets, the method exhibits better performance than other methods for many data sets, and found its own significance of existence in the field of regression.

1 Introduction

We consider discovering a numerical relationship from data. Multiple linear regression (MR) or Hayashi's QT method [1] are widely used to find linear relationship among variables. There has been work on mixture models which use more than one regression functions to fit data. Mixture models can be classified into hard and soft. Hard mixture uses only one function to obtain an estimate, while soft mixture uses multiple functions to construct an estimate. CART [2] and RF6.4 [3] are classified as hard, while HME [4], and MARS [5] are classified as soft. In terms of readability, hard mixture is superior to soft mixture.

Employing the hard mixture framework, this paper proposes a method called *nominally conditioned multiple regression (nc-MR)* for finding a set of regression rules to fit multivariate data containing both numerical and nominal variables.

2 Nominally Conditioned Multiple Regression : nc-MR

An observation is given in the form of $(q_1, \dots, q_K, x_1, \dots, x_J, y)$, where q_k is a nominal explanatory variable, x_j is a numerical explanatory variable, and y is a numerical criterion variable. For each q_k we introduce q_{kl} defined as follows: $q_{kl} = 1$ if q_k matches the l -th category, and $q_{kl} = 0$ otherwise.

To explain the behavior of y by using $\mathbf{q} = (q_1, \dots, q_K)$ and $\mathbf{x} = (x_1, \dots, x_J)$, we consider regression rules as shown in eq. (II), where a conditional part defines the nominal subspace where a linear ϕ is to be applied.

$$\text{if } \bigwedge_k \bigvee_{q_{kl} \in Q^i} q_{kl} \text{ then } y = \phi(\mathbf{x}; \mathbf{w}^i), \quad i = 1, \dots, I \quad (1)$$

We consider the implementation of eq. (II) using a four-layer perceptron. To express the conditional part numerically, the following c is introduced.

$$c(\mathbf{q}; \mathbf{v}^i) = \sigma \left(\sum_{k=1}^K \sum_{l=1}^{L_k} v_{kl}^i q_{kl} \right) \quad (2)$$

Here \mathbf{v}^i is a vector of weights $\{v_{kl}^i\}$, and $\sigma(h)$ is a sigmoid function. Given appropriate weights, the following can approximate the final output y defined by eq. (II) with enough accuracy.

$$F(\mathbf{q}, \mathbf{x}; \mathbf{v}^1, \dots, \mathbf{v}^I, \mathbf{w}^1, \dots, \mathbf{w}^I) = \sum_{i=1}^I c(\mathbf{q}; \mathbf{v}^i) \phi(\mathbf{x}; \mathbf{w}^i) \quad (3)$$

If R is given properly, eq. (3) can be exactly represented by the following, which can be implemented by a four-layer perceptron.

$$f(\mathbf{q}, \mathbf{x}; \boldsymbol{\theta}) = \sum_{j=0}^J c_j x_j, \quad c_j = \sum_{r=1}^R v_{jr} \sigma_r, \quad \sigma_r = \sigma \left(v_{r0} + \sum_{k=1}^K \sum_{l=1}^{L_k-1} v_{rkl} q_{kl} \right) \quad (4)$$

We extract a set of regression rules from a learned four-layer perceptron. Let \hat{v}_{jk} , \hat{v}_{rkl} and \hat{v}_{r0} be the weights obtained after learning of the perceptron. Then, the coefficient c_j^μ of variable x_j for observation μ is calculated as follows:

$$c_j^\mu = \sum_{r=1}^R \hat{v}_{jr} \hat{\sigma}_r^\mu, \quad \hat{\sigma}_r^\mu = \sigma \left(\hat{v}_{r0} + \sum_{k=1}^K \sum_{l=1}^{L_k} \hat{v}_{rkl} q_{kl}^\mu \right) \quad (5)$$

Coefficient vectors $\mathbf{c}^\mu = (c_0^\mu, c_1^\mu, \dots, c_J^\mu)$ can be compressed by VQ (vector quantization) into $I (\ll N)$ representative vectors $\mathbf{a}^i = (a_0^i, a_1^i, \dots, a_J^i)$ for $i = 1, \dots, I$.

Next we define the labeling function, $i(\mathbf{q}^\mu) = \arg \min_i \|\mathbf{c}^\mu - \mathbf{a}^i\|^2$ which returns the cluster number corresponding to μ . Using $i(\mathbf{q})$, we get the following:

$$\text{if } i(\mathbf{q}) = i \text{ then } \hat{f} = \sum_{j=0}^J a_j^i x_j, \quad i = 1, \dots, I \quad (6)$$

Finally, we transform the function $i(\mathbf{q})$ into a conjunctive normal form to obtain regression rules as defined by eq. (II). For this purpose, we solve a classification problem with $\{(\mathbf{q}^\mu, i(\mathbf{q}^\mu)), \mu = 1, \dots, N\}$ as training data.

We have to find the proper model parameters: R^* in eq. (II) and I^* in eq. (6). Here we employ the Bayesian Information Criterion (BIC) [6].

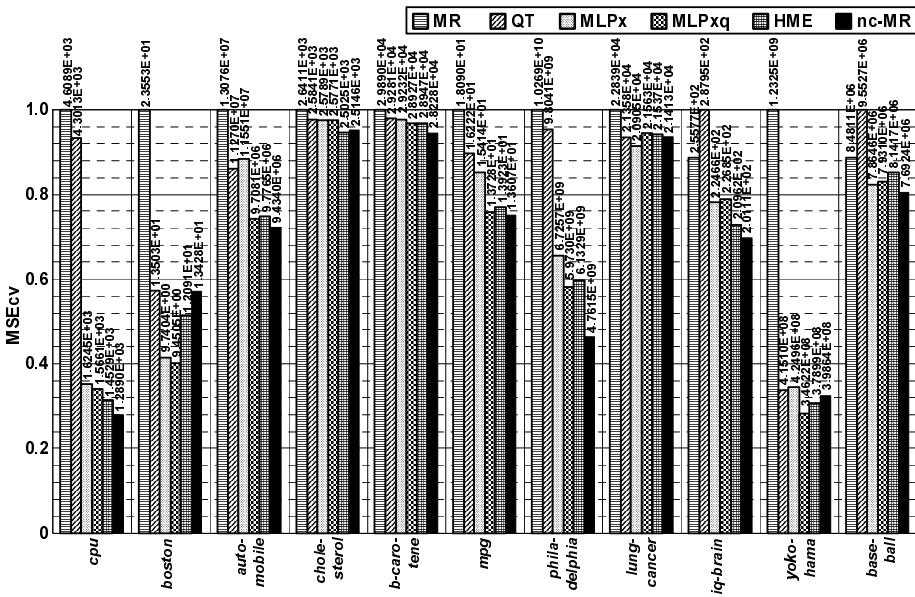
3 Experiments and Consideration

We evaluated the performance of nc-MR using 11 real data sets¹. Table II outlines the data sets. Here, K_{all} denotes the total number of categories.

¹ Cpu, Boston and automobile data were from the UCI MLDB. Cholesterol, b-carotene, mpg, Philadelphia, lung-cancer and iq-brain data were from the StatLib. Yokohama data was from the official web page in Japan. Baseball data was from the directory of Japanese professional baseball player in 2005.

Table 1. Real data sets used in the experiment

data set	contents of criterion variable	N	J	K _{all}
cpu	performance of CPU	205	6	26
Boston	housing price in Boston	486	12	76
automobile	price of cars	169	13	20
cholesterol	amount of cholesterol	297	5	13
b-carotene	amount of beta-carotene	315	10	6
mpg	fuel cost of cars	388	5	28
Philadelphia	housing price in Philadelphia	99	3	5
lung-cancer	survival time of lung cancer patients	126	3	6
iq-brain	intelligence quotient	20	5	3
Yokohama	housing price in Yokohama	558	4	23
baseball	annual salary of Japanese professional baseball players	211	6	14

**Fig. 1.** Comparison of 10-fold cross-validation error (real data)

First, we performed 100 times of perceptron learning, where R was changed from 1 to 5. The $BIC(R)$ was minimized most frequently when $R=2$ or 3. Next, using learning results of the perceptron having R^* , we quantized coefficient vectors with $I=1,2,\dots,20$. Using the C4.5 program, we got the final regression rules.

We compared the performance of nc-MR with other methods: MR(linear regression), QT(linear regression using both numerical and nominal variables) MLP(3-layer perceptron), and HME. Solution quality was measured by 10-fold CV errors. Figure 1 shows the best results for each method. We can see that nc-MR exhibited the best performance for 7 data sets. Although nc-MR could not exceed MLP for 3 data sets, nc-MR is superior to MLP in terms of readability.

We compared the CPU time of the above methods. Obviously MR and QT were the fastest due to the linearity, and MLPx was rather fast since it uses only numerical variables. The nc-MR was faster than HME for 8 data sets, and was slower than MLPxq because nc-MR needs rule restoration after learning.

We consider classifying data into 4 groups in the context of regression. Group 1 is characterized as *numerically linear*, where MR can fit very well, and nominal variables do not play any significant role. Cholesterol and b-carotene data sets belong to Group 1. Group 2 is characterized as *numerically and nominally linear*, where QT fits very well while MR does not. Lung-cancer and Yokohama data sets belong to Group 2.

Group 3 is characterized as *nominally conditioned linear*, best suited to the proposed nc-MR. Six data sets such as cpu, automobile, mpg, Philadelphia, iq-brain and Baseball data belong to Group 3. We suppose many data sets may belong to Group 3.

Finally, Group 4 is characterized as *strongly nonlinear*. Boston data set belongs to this group. MLP and nominally conditioned polynomial regression [3] nicely fit Group 4.

References

1. Tanaka, Y.: Review of the methods of quantification. *Environmental Health Perspectives* 32, 113–123 (1979)
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Chapman and Hall, Boca Raton (1984)
3. Tanahashi, Y., Nakano, R.: Bidirectional clustering of MLP weights for finding nominally conditioned polynomials. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009. LNCS, vol. 5769, pp. 155–164. Springer, Heidelberg (2009)
4. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and EM algorithm. *Neural Computation* 6(2), 181–214 (1994)
5. Friedman, J.H.: Multivariate adaptive regression splines. *The Annals of Statistics* 19(1), 1–141 (1991)
6. Schwarz, G.: Estimating the dimension of a model. *Annals of Statistics* 6, 461–464 (1987)

A Novel Continuous Dual Mode Neural Network in Stereo-Matching Process

Lukasz Laskowski

Czestochowa University of Technology, Department of Computer Engineering,
Al. A.K. 36, 42-200 Czestochowa, Poland

Abstract. In the present paper we describe completely innovative of architecture of artificial neural network based on Hopfield structure. It is analogue implementation of dual mode Hopfield-like network for solving stereo matching problem. Considered network consists of basic layer of neurons realized by analogue Hopfield-like network and managementing layer. Thanks to the using of managementing layer there is a possibility of modification of the connection weights between the neurons in basic layer. This enables of improvement and correction of the solution. In the present article we also describe energy function for basic layer and the condition of syntactic correctness of the solution, allowing to correct connection weights. The network considered here was taken under experimental tests using real stereo pictures as well simulated stereo images.

Keyword: Hopfield, stereovision, depth analysis, dual mode network.

1 Introduction

People's mobility is possible to a large degree thanks to sight. Most of the information required for orientation and mobility is gathered through the visual channel. People, who lost or damaged this sense are not able to live independently. In order to help people who are blind or visually impaired scientists makes an effort to develop devices, assisting in independent mobility. One of solution can be portable binocular vision system, which makes possible in orientation and mobility for blind users. The system can be based on stereovision [1]. The advantages of stereovision include ease of use, non-contact, non-emission, low cost and flexibility. For these reasons, it focuses attention of scientists to develop of stereo-vision methods.

In the present paper, a completely innovative architecture of network to solving stereo matching problem [2, 3] is described – Continuous Dual Mode Neural Network (CDMNN). Due to dual layer architecture, the network presented has a possibility of correction of solution and elimination of false matching. This leads to improving of efficiency of network's working efficiency (significantly decreased error). In the case of classical Hopfield-like networks [5, 6] it is often impossible to make all assumption of solution in energy function (for example correspondence of edge in depth map to edge in image). Dual mode networks give us this the possibility to check the solution and weight correction. A solution like this

has never been used in stereo matching problems so far. The confrontation of presented here Continuous Dual Mode Neural Network with classical Hopfield-like network clearly indicates the superiority of CDMNN (both network was working on the same energy functions and in the same conditions). After a single modification of interconnection weights by management layer, an error falls half as much, which indicates the advisability of this process.

2 Architecture of Analogue Dual Mode Neural Network

Author's structure was inspired by architecture proposed by S. Lee and J. Park in [7, 8]. Original dual mode neural networks was not fitted to solve stereo matching problem, mainly because of using discrete Hopfield network in basic layer. The structure presented here is further modification of this kind of network. That is the reason why the authors decided to use only the main architecture of dual mode networks, modifying of layers operation. Architecture of this structure was depicted in fig. 1. As can be seen, this kind of structure consists of two kinds

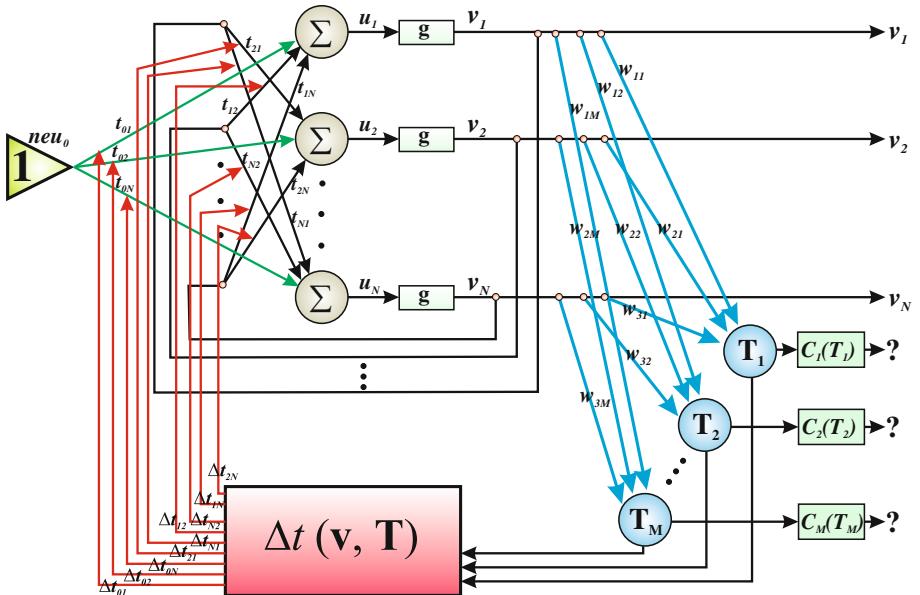


Fig. 1. The architecture of dual mode neural network

of neurons: neurons in basic layer and neurons in management layer. The function of these two kinds of neurons is completely different. The additional neuron neu_{00} is still active (potential equal 1) and its role is to supply external currents to neurons in basic layer.

3 Experimental Results

The results of stereo matching process carried out by CDMNN can be seen in fig. 2.

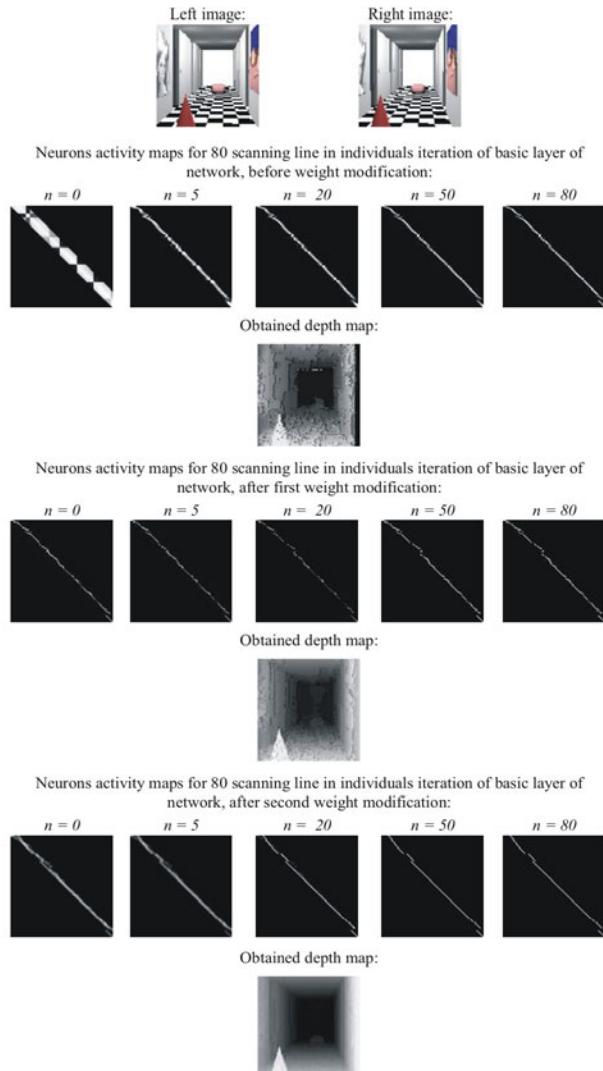


Fig. 2. The result of stereo matching process carried out by CDMNN for simulated stereo-images

At figure 2 in the first row stereo pictures, used for stereo matching process was shown. The second row shows neurons activity maps for 80 scanning line (arbitrary assumed) in iterations (number of "n") of basic layer of CDMNN. The neurons activity map is helpful to the analysis of network working. It can be interpreted as a graphical form of fitting matrix to investigated line. In next row obtained (by basic layer of CDMNN) the depth map can be seen. The same sequence (instead of stereo-pictures) was repeated after the modification of connection weights in basic layer.

The analysis of first running of basic layer for simulated pictures (fig. 2), numerous mistakes of fitting can be seen. The relatively high error (29.49%) was obtain. On the basis of output depth map, management neurons are calculating the connection weight's corrections and adjust it towards minimization of Dual Function. After the modification, basic layer is activated with new interconnections strengths. In this running the error amounts to 11.45%, which is significantly less then with no connection modification. After the third running the solution obtained is much better, then in the previous running. The relative error amounts only to 4.37%. The stereo matching seems to be unique and disparity is continuous in areas. the sequence of disparity is also kept.

4 Conclusion

The outcomes of investigating of this kind of neural structure lead to the conclusion that the weights correction based on the obtained solution is essential to reach a correct solution of stereo correspondence problem. This approach can be crucial to applications requiring really high accuracy (e.g. system of supporting visually impaired people).

References

- [1] Aleksander, I.: Artificial vision for robots. Korgan Page (1983)
- [2] Barlow, H.B., Blackmore, C., Pettigrew, J.D.: The natural mechanism of binocular depth discrimination. *J. Physiology* 193, 327–342 (1967)
- [3] Faugeras, O.: Three-dimensional computer vision. In: A Geometric Viewpoint. MIT, Cambridge (1993)
- [4] Alvarez, L.: Dense Disparity Map Estimation Respecting Image Discontinuities: A PDE and Scale-Space Based Approach. *Journal of Visual Communication and Image Representation* 13, 3–21 (2002)
- [5] Hopfield, J.J., Tank, D.W.: “Neural” computation of decisions in optimization problems. *Biological Cybernetics* 52, 141–152 (1985)
- [6] Hopfield, J.J., Tank, D.W.: Artificial neural networks. *IEEE Circuits and Devices Magazine* 8, 3–10 (1988)
- [7] Lee, S., Park, J.: Dual-mode dynamics neural network (D2NN) for knapsack packing problem. *Neural Networks, IJCNN* 3(25-29), 2425–2428 (1993)
- [8] Lee, S., Park, J.: Dual-mode dynamics neural networks for combinatorial optimization. *Neurocomputing* 8, 283–304 (1995)

Learning Invariant Visual Shape Representations from Physics

Mathias Franzius and Heiko Wersing

Honda Research Institute Europe GmbH

Abstract. 3D shape determines an object’s physical properties to a large degree. In this article, we introduce an autonomous learning system for categorizing 3D shape of simulated objects from single views. The system extends an unsupervised bottom-up learning architecture based on the slowness principle with top-down information derived from the physical behavior of objects. The unsupervised bottom-up learning leads to pose invariant representations. Shape specificity is then integrated as top-down information from the movement trajectories of the objects. As a result, the system can categorize 3D object shape from a single static object view without supervised postprocessing.

1 Introduction

Invariant shape recognition is a hard problem because many three-dimensional objects undergo extreme appearance variations when they rotate in-depth or light conditions change. Slowness learning can be used to learn this invariance: even if views of the same object are very different, they more often occur in close temporal relationship during object interaction than views of distinct objects [17]. Such models allow object identification since different views of the same object form clusters in the learned feature space and views of visually distinct objects typically cluster in distinct regions. However, objects of similar categories do not generally appear more often in close temporal relationship than objects of different categories. The relative distances of object clusters thus remain undetermined after SFA learning, and minimal noise can cause object clusters to permute their positions in different simulations. Here, we extend a model for invariant object recognition such that a small autonomously generated additional input signal determines the relative positions of object clusters and thus implements a meaningful shape similarity measure. Although in general the problem of deriving 3D object shape from a single view is ill-defined, humans can often already guess an object’s shape from a single view and predict how it would move when it is agitated. We demonstrate here a model of this behavior for a limited object set.

2 Methods

Twelve objects of four distinct shapes (cone, cube, sphere, capsule) and three colors (red, green, blue) were dropped into a box with tilted ground plane and their

behavior was simulated with a physics engine. In each time step, a bounding box around the object was estimated from the visual data and a quadratic section of 50×50 RGB pixels containing the object was cut out. Thus, the visible transformations of the object consisted of in-depth and in-plane rotations, scaling, small positional jitter, and change of lighting direction. After dropping an object onto the surface, 400 video frames were recorded with a framerate of 10Hz, followed by an all-black image after which the process repeats with another object dropping into the box. The process was repeated until a total of 100,000 views and 250 trajectories, corresponding to $2\frac{3}{4}$ hours, were generated.

Two properties of the moving objects were measured: the total time until they come to a stop and the distance traveled downhill. From these 250 two-dimensional trajectory descriptors 120 were randomly selected and classified by k-means ($k = 4$). The resulting trajectory clusters T coincide to a large degree with object shape and thus the fact that the trajectories of two objects falls into the same trajectory cluster can be used as a learning signal. Subsequently, each training object trajectory was classified as belonging to one of these clusters. Proportionally to the co-occurrence of two objects (A, B) in T , additional randomly selected view pairs (V_A, V_B) were presented to the system after the first unsupervised learning phase. Thus, the system was additionally trained with sequences of object pair views, which are likely to have similar shape. While all views of the object movies were weighted equally during training, the learning weight was increased by a factor of 10 for the additional shape training views.

The slowness objective was optimized using Slow Feature analysis (SFA) [9]. As the image dimensionality of 7500 is too large to perform nonlinear SFA in a single step, we employed a hierarchical network architecture as described in [3].

For analyzing learned representations, we performed unsupervised k-means clustering. Afterwards, the feature representations of all training views are assigned to the closest cluster center. Since k-means randomly permutes cluster identities in each run, we always identify the permutation of cluster names with the highest overlap to ground truth shape categorization.

3 Results

First we analyze the learned representations after training the hierarchical network with videos of objects dropped into the box with a tilted floor without any further shape-related training. For this purpose, we compute the average values (i.e., the cluster centers) of each object in the slowest ten components in the highest layer of the hierarchical network. The left panel of Fig. II shows the distance matrix between all pairs of clusters after normalizing the highest distance to 1. As expected, most clusters are roughly equidistant (with the exception of the cone clusters). These distances fluctuate for repeated simulations due to the small amounts of noise injected into the hierarchy. In this representation, a view of a green cube, for example, is on average as similar to a red cube as to a red sphere. Except for the distances between cones, there is no evident clustering of views from objects with same shape or color.

In a second step, we investigate how well the distance matrix of object view clusters can be “programmed” explicitly when only correct pairs of views of objects from the same shape category are shown. For this purpose, 100 views of each shape training view pairs (i.e., {(red,green), (red, blue), (green,blue)} $\times\{\text{cone, cube, sphere, capsule}\}$) were presented to the system additionally to the videos of the dropped objects. The central panel of Fig. I shows the resulting distance matrix. Here, all clusters of views of objects with identical shape are very close to each other, whereas all cross-shape cluster distances are roughly equal and close to the maximum distance. As desired, in this representation, views of objects of the same shape are categorized as similar and views of objects with distinct shapes are considered distinct. This observation is quantified by performing k-means ($k=4$) clustering of the view feature representations and assigning each view to the closest cluster center. On average over 20 trials, 99.5% of object views were assigned to the correct shape cluster.

Finally, we characterize the full system with unsupervised top-down learning. As before, all layers are trained with the movies of objects dropped into the box. Again, the movement trajectories of all objects are measured and clustered using k-means ($k=4$). More than 90% of the trajectories cluster consistently with a shape class. After training all layers of the network with the dropped object movies, pairs of object views are presented with a likelihood proportional to the frequencies of co-occurrence of objects in the trajectory clusters T . The right panel of Fig. I shows the resulting distance matrix after the presentation of 400 such randomly selected pairs. Again, we perform k-means ($k=4$) clustering of the view feature representations and assigning each view to the closest cluster center. Similar to the results with the additional supervised training views above, this autonomously learned representation categorizes views by object shape. On average, 98.0% of all views are categorized as belonging to the same shape, and, on average, less than 100 view pairs are sufficient to reach 90% shape categorization performance. Note that this number is less than one view pair per object pair ($12 \times 12 = 144$) and that many presented view pairs show the same object (i.e., same color and same shape).

For comparison, we quantified the sizes of shape and color clusters in the pixel space. The average cluster diameter of all views of objects with the same color is 5.9 times smaller than the average size of clusters of all views of objects with identical shape. Performing 100 times k-means clustering ($k \in \{3, 4\}$) in the raw pixel space never achieved a higher overlap than 82% with either shape or color clusters in 100 trials but on average the overlap with the color clusters was 41% larger.

4 Discussion

We have presented a system for unsupervised learning of a visual feature representation that clusters views of objects with similar 3D shape. The system learns invariance to pose variations of the objects and color variations of objects within categories. The shape information is autonomously derived from the movement

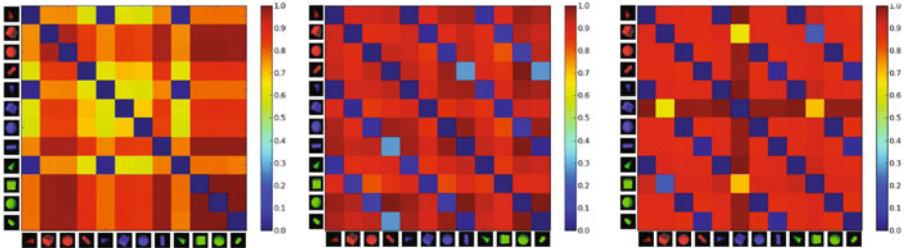


Fig. 1. Cluster center distances. Representations of single object views cluster in the learned feature space. Each subplot depicts normalized pairwise distances between the cluster centers of any given object in the learned features space. Left: Without any inter-object similarities learned (i.e., completely unsupervised), cluster center distances are undetermined and change between simulations. Center: After additional supervised presentation of 100 view pairs between objects of similar shape, features of objects with similar shape cluster but keep a similar distance to unrelated shape clusters. Right: Based on the similarity of their physical movement trajectories (“self-supervised”), most features of objects with similar shape cluster as in the supervised case.

behavior of the objects when dropped onto a tilted surface. Thus, shape information gets directly integrated into the learned visual feature representation.

We have shown that without additional self-derived training views, most object representations cluster roughly equidistantly, i.e., they are distributed as localized clusters on a hypersphere in the learned feature space. Although the SFA optimization itself is deterministic and guaranteed to find the globally optimal solution, the positions of the object clusters permute in different simulations under the addition of a small amount of noise. Theoretically, a single mini-sequence consisting of one pair of views between two distinct objects A and B already leads to a symmetry breaking such that the clusters of A and B have a smaller distance to each other than any other cluster pair. This sensitivity to small perturbations is an ideal basis for learning from few examples, either supervised (as shown in the central subplot of Fig. II) or autonomously (as shown in the right panel of the same figure). Practically, some intermediate visual cues from the visual hierarchy will likely bias these cluster distances in simulations, even when no temporal inter-object relationship has been experienced during training (left panel of Fig. II). However, as we have shown, object shape can not trivially be clustered in the pixel space. Instead, object color is a more prominent cue. Thus, there is a bias in the input space to cluster by color and not by shape and the fact that the resulting representations become color-invariant and shape-specific demonstrates that the input categorization bias is small and can be “overwritten” even with few examples. Additionally, we have shown some robustness to noise in the self-derived additional shape training views, since the trajectory clustering does not perfectly coincide with the shape clusters.

This work is related to slowness-based invariant object recognition. While invariant object recognition with much more complex objects has been shown earlier [13], we restrict the object complexity here to four shapes and three colors

for the sake of simplicity. While it seems likely that our system can find invariant representations for visually more complex objects, ground truth shape categories are less evident and thus objective evaluation is harder. However, one shape representation could be evaluated as better than another if it facilitates a given task. Such an integrated approach is an interesting subject for further research. The main difference to existing slowness-based invariant object recognition systems of our approach is the systematic integration of top-down cross-object similarity of discrete objects, specifically for learning 3D shape categories. A similar hierarchical model architecture has earlier been shown to model most known functional aspects of hippocampal spatial codes (i.e., place cells, head direction cells, spatial view cells and grid cells [2]). As the hippocampus is crucial as a memory hub and well-known for time-delayed replay of previous experiences [4], we hypothesize that if replay sequences of objects with similar movement trajectories occur more often than those of different shape, i.e., out of experienced temporal context but in new task-specific context, the hippocampus could implement a mechanism similar to the one proposed here.

Additionally, this work is related to affordance learning approaches from the developmental robotics community [5,6,8]. These approaches also show autonomous learning of affordances but use sophisticated robotic actuators, whereas we have shown our approach only for simulations. However, these approaches tend to employ much simpler visual features (e.g., nearest neighbor classification in the pixel space), whereas our approach focuses on the learning of visual feature representations with invariances to strongly changing visual stimuli.

References

1. Einhäuser, W., Hipp, J., Eggert, J., Körner, E., König, P.: Learning viewpoint invariant object representations using a temporal coherence principle. *Biol. Cyber.* 93, 79–90 (2005)
2. Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness lead to place, head-direction and spatial-view cells. *PLoS Comp. Biol.* 3(8), e166 (2007)
3. Franzius, M., Wilbert, N., Wiskott, L.: Invariant object recognition with slow feature analysis. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 961–970. Springer, Heidelberg (2008)
4. Gupta, A., van der Meer, M., Touretzky, D., Redish, A.: Hippocampal replay is not a simple function of experience. *Neuron* 65(5), 695–705 (2010)
5. Metta, G., Fitzpatrick, P.: Better vision through manipulation. In: Proc. 2nd Inter. Workshop on Epigenetic Robotics, vol. 11, pp. 109–128 (2002)
6. Ridge, B., Skočaj, D., Leonardis, A.: A system for learning basic object affordances using a self-organizing map. In: Proc. ICCS (2008)
7. Rolls, E.T., Stringer, S.M.: Invariant visual object recognition: A model, with lighting invariance. *Journal of Physiology - Paris* 100, 43–62 (2006)
8. Stark, M., Lies, P., Zillich, M., Wyatt, J., Schiele, B.: Functional object class detection based on learned affordance cues. In: Gasteratos, A., Vincze, M., Tsotsos, J.K. (eds.) ICVS 2008. LNCS, vol. 5008, pp. 435–444. Springer, Heidelberg (2008)
9. Wiskott, L., Sejnowski, T.: Slow feature analysis: Unsupervised learning of invariances. *Neural Comp.* 14(4), 715–770 (2002)

Algorithms Creating Algorithms

Boyan Lalov

Department of Electrical Engineering, Electronics and Automatics,
Technical University - Sofia, 1000 Bulgaria
boyanlalov12@abv.bg

Abstract. The spontaneous synthesis of algorithms in artificial neural networks is experimented and analyzed on three hierarchical levels and there are three basic problems: How the engram of memory connects the neuron state and the combination of messages from other neurons. How the cortical column decodes the parallel series of impulses from other columns. How the sensory and motor anticipation processes interact to construct the branches of a new algorithm.

Keywords: engrams of memory, self-learning by random steps, sensory and motor anticipation, decoders of space-temporal features, algorithm aggregation.

1 Engrams and Proto-ensegrams

We call proto-ensem the early postnatal molecular lock of the neuron which reacts to a specific combination of activated inputs. It is possible that the transformation of input impulses in a mediator combination and then in a chemical compound as an equivalent of the message starts or stops the postsynaptic impulse generation. The proto-ensegments indicate the features of the most frequent standard patterns of the environment.

Our program simulates a process of self-organization. It selects *random but often appearing fragments* of the receptor matrix to make the connections of the network. In other words, the output signals of the receptor matrix are going upward and make groups of specific combinations of connections from receptors to responding neurons (fig.1). The proto-ensegments in these neurons indicate less or more complex features, respectively. The combinations of signals, coming from higher level to low level neurons (top down connections), give information about more complex features. An engram appears when a neuron of low level records which state it has (0 or 1) and which is the specific combination of active connections sending messages from the higher level to the neuron. In other words, the neurons of a lower level, depending on their state, record *positive or negative engrams*. These engrams respond to specific combinations of active inputs. In the future the same combination of messages (features) may provoke the same state of the neuron. If the stream of information coming from the receptor matrix is discontinued for a while then the top-down connections may excite or inhibit the first level neurons. It depends on the influence of the positive or negative engrams. (Lalov, Patent 28383/1980, BG)

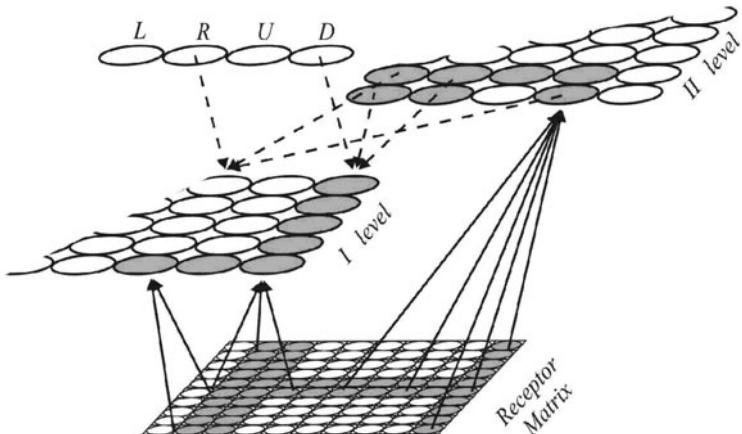


Fig. 1. Upward and top down connections

In our computer simulations the top-down connections can restore previous states of the low level neurons when a new pattern on the receptor matrix has some known features. Then the higher level neurons respond to these features and provoke through the top-down signals the appearing of an old pattern from the past in the first level neurons.

2 Sensory and Motor Anticipation

Sensory Anticipation. On fig. 1 the signals from motor neurons R and D (Right and Down) are provoking a shift of the pattern on the receptor matrix. The combination of sensorial messages from the second level (concerning the initial state of the matrix) and the R and D messages from the motor zone are recorded in the first level neurons (indicating the state after the shift). If the same combination of messages appears in the future, the neurons of the first level will predict (by changing their state) that the pattern on the receptor matrix will move to the same direction.

Motor Anticipation. To predict the necessary motor activity, which leads to a desirable state of the environment the network prepares records in the adequate motor neurons: *Random motor neurons are activated* to move patterns on the receptor matrix. The network makes records in these motor neurons (fig. 1). These records are the result of the combination of messages coming from the high level neurons (concerning the initial position of the pattern) and messages coming from the low level neurons, signaling the position after the shift. When such combination is being read in the future the motor neuron that receives the proper information will send a signal to carry out the correct movement. With such combinets - combinative networks we have experimented simple constructive

and marsh route motions. By random connections and random movements we have provoked processes of self-learning in: a) pattern recognition by restoring of damaged patterns; b) association of symbols and images (verbal - non verbal association); c) arranging a construction according to a given sample.

When a combination of features appertains to many different patterns, it does not give much information. The unique features are the most valuable. Using unique features we can minimize the necessary number of neurons and connections in the network. Our simulating program uses neurons that indicate the absence of known features. This is important information. The experiment proved that such neurons help to recognize the patterns much better.

3 Cortical Columns: Decoders of Space-Temporal Features

How does the cortical column decode the actual local stream of parallel series of impulses and transforms it into a specific series of impulses and pauses of a long axon cell? Each neuron from the constellation around a long axon (central) cell of the cortical column can read (identify) by its proto-gram a familiar combination of impulses coming simultaneously to its inputs. Different messages coming into the column pass different routes of neurons (fig.2). The impulses produced from these neurons arrive, more or less, with various interval of delay before getting into the long axon (central) cell. The combination of messages reaching at last this central cell makes it produce a specific code - a consequence of impulses and pauses. Thus, the impulses and pauses generated by the central cell are a final reaction summarizing the initially incoming messages. Due to the neuron retardation some simultaneous messages can arrive consecutively to the central cell and vice versa - some consecutive messages can arrive simultaneously.

Let us analyze a simple computer game in which our artificial neural network takes part. In a matrix with black and white squares the squares are situated randomly. The columns of the matrix are moving downwards. There is a *grey square* in the lower part of the matrix. Its movements - to the left, to the right and upwards, controlled by respective motor neurons, are avoiding collision with the black squares moving downwards. Initially the motor neurons control the grey square *by random activation* and it makes 1, 2 or 3 steps depending on the activated neurons. The sensory neurons are grouped in cortical columns analyzing the space-temporal features of different local areas of the matrix. The cortical columns register the movements of the grey square and the black squares. The signals are processed as it has been described above. The long axons of the cortical columns connect the active motor neurons and *make positive engrams each time when the collision is avoided*. After a period of self-learning the columns which register the actual area of the grey square can control its successful steps without collision. The experiment showed that by activation of random motor neurons the neural network learns to avoid the collisions and moves the grey square upwards.

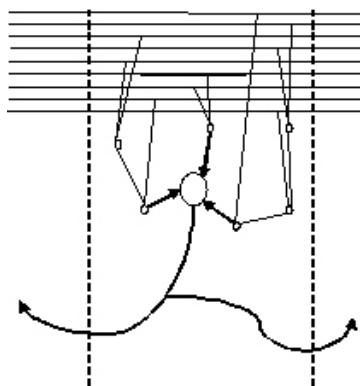


Fig. 2. A schematic presentation of connections between trans-cortical pathways and a cortical column. Thanks to the neuron retardation some simultaneous messages can arrive consecutively to the central cell and vice versa - some consecutive messages can arrive simultaneously.

4 Generalization

By simple simulations we tried to reveal the mechanisms, which code the afferent information and transform it into temporal patterns of impulses and pauses that can provoke the necessary effectors reaction. These temporal patterns can control and coordinate the activity of eye and arm muscles, by means of which the brain follows the contours, investigates and moves the objects. Another case of such a transformation is the imitation activity, playing a fundamental role in the process of learning, or human speech transforming the space models of events in verbal activity and vice versa, etc. In the early postnatal period of development the variety of electrochemical pulsation of the pathways of millions of neural fibers, which pass in the close proximity of the given columns, allows the creation of definite connections of the nerve pathways with the neurons of these columns due to the multiple repetitions of some typical series. The central (long axon) cells give generalized answers indicating some definite spatial and temporal properties of certain sensorial events in the cortex. In fact, there may be some ambiguity in a central cell answer because sometimes different events can give equal temporal pulse patterns. However, this ambiguity does not stand in the way of the general activity of the cortex. The cortical column is multifunctional and its output messages may have determined meaning depending on the actual signals of other columns. This principle of organization is used by the nervous system to send messages to other (higher) levels in the cortex and to make cascades of groups of columns. The information processes in the cortex carry out routine forms of behavior and take place as an automatic interaction between the sensory and the motor cortex. But the main point of the problem is how the new forms of behavior are created.

5 Creating Fragments of Algorithms

On fig.3 three basic blocks of the brain cortex are presented: S - Sensory, M - Motor and P - Prefrontal. The Prefrontal cortex (P) is built over the (pre)motor cortex but its structure is similar to the associative sensory cortex. The Prefrontal cortex has highly developed upper layers and reverberating circuits. It serves to accumulate and maintain complete sensory information concerning the integral conditions during the period of solving of current non routine problems. It is mostly a system for temporary storing of information about actual events, used for the construction of new forms of behavior. As it has been mentioned in paragraph 3, the sensory anticipation is the appearance in the sensory cortex of models of probable (expected) changes in the environment. When the actual conditions of the environment are loaded in P, and in the Motor block M there is an activation of the model of the intended movement, then the Sensory block S occupies a state responding to the probable changes in the environment. How is this process of interaction carried out on the level of engrams and how were such useful engrams obtained in the past? In similar conditions and with experimental or even random actions (orientation activity) in the past, the features of the reality were changed in the sensory block S. Under the simultaneous influence of the signals coming from the blocks P and M, there (in S) were created engrams (molecular locks) that will react to similar conditions and actions. In the future all signals similar to such intended actions and current conditions will activate these engrams and the respective neurons in S. Thus, an interpolation or it would better to say a superposition of features emerges. This process has the character of a prognosis, of Expectation. It is a fundamental act of cognition. However, by acting we do not prepare solely engrams of sensory anticipation in S where the model of the result appears (fig.3). Meanwhile, the pattern of the conditions is maintained in P and the combined messages from P and S will be recorded in M as engrams of the action that provokes the result in S.

Owing to these records, the networks of the Motor cortex will anticipate the necessary actions in M if similar conditions are created in P, when the expected result (this time as an aim) occupies the sensorial region S! This is a process of activation or extraction of necessary elements by reading the motor long term memory. Based on these elements a model of the motor solution will be

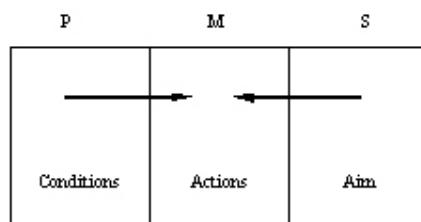


Fig. 3. Activation of the neural model of the necessary actions under the influence of the maintained models of the aim and conditions

constructed. The anticipation of the necessary motion is the other fundamental act of cognition. In our opinion, there is another mechanism carrying out subtle comparison and correction in the motor cortex. The momentary appearance of the aim in S, quickly changed by the pattern of reality, may periodically provoke sensorial changes accepted and processed by the columns of M, decoding the space-temporal features of the difference. Each time when the reaction of M eliminates this difference and reality corresponds to the pattern of the aim, the model of the successful motor activity is memorized. Such differentiating networks which register non-corresponding features are products of juxtapositions between real and anticipated in the early stages of neural network development.

6 Conclusion

We have tried to delineate the contours of the ABC theory (Algorithms of Brain Creativity), which describes the structure of the brain cortex as an aggregate for new forms of behavior. The synthesis of a new complex motor program is carried out through numerous cyclic attempts for a preliminary construction of separate branches. In practice, the aim is decomposed into preliminary aims by means of a sequence of motor and sensory anticipations. That is to say - we move from the stem of the tree of aims to the branches. Sub-aims or sub-sub-aims will be canceled or changed when there is a lack of a proper solution. Successful intermediate solutions (coincidences of expectations and results) are memorized as sensory and motor engrams. The solving of the problem is repeatedly taken back to the beginning - the process is converged to the main aim by itself. The search of a solution is similar to a random process, but in fact this is an algorithm creating algorithms. The fulfillment of the complex aim is possible if there are successful anticipations - sensory and motor results confirmed by practice. After the synthesis of a new fruitful algorithm, P stops interfering in the already routine behavior of the sensory-motor cortex and waits until new non-routine activities appear.

Assessing Statistical Reliability of LiNGAM via Multiscale Bootstrap

Yusuke Komatsu¹, Shohei Shimizu^{1,2}, and Hidetoshi Shimodaira¹

¹ Dept. Math. and Comp. Sci., Tokyo Institute of Technology, Japan

² The Institute of Scientific and Industrial Research, Osaka University, Japan

Abstract. Structural equation models have been widely used to study causal relationships between continuous variables. Recently, a non-Gaussian method called LiNGAM was proposed to discover such causal models and has been extended in various directions. An important problem with LiNGAM is that the results are affected by the random sampling of the data as with any statistical method. Thus, some analysis of the confidence levels should be conducted. A common method to evaluate a confidence level is a bootstrap method. However, a confidence level computed by ordinary bootstrap is known to be biased as a probability-value (*p*-value) of hypothesis testing. In this paper, we propose a new procedure to apply an advanced bootstrap method called multiscale bootstrap to compute *p*-values of LiNGAM outputs. The multiscale bootstrap method gives unbiased *p*-values with asymptotic much higher accuracy. Experiments on artificial data demonstrate the utility of our approach.

Keywords: bootstrapping, reliability, structural equation models, independent component analysis, non-Gaussianity.

1 Introduction

Structural equation models [1] have been widely applied to analyze causal relationships in many fields. Many methods [2,3] have been developed to discover such a causal model when no prior knowledge on the network structure is available. Recently, a non-Gaussian method called LiNGAM [4] was proposed. The new method estimates a causal ordering of variables using passive observational data alone. The estimated ordering is correct if the causal relations form a linear structural equation model with non-Gaussian external influence variables and the sample size is *infinitely large*. In practice, however, the sample size is finite. The finite sample size induces statistical errors in the estimation, and the estimated ordering may not be right even when the model assumptions are reasonable. Thus, some analysis of the statistical reliability or confidence level of the estimated ordering should be done.

A common procedure to evaluate such a confidence level is statistical hypothesis testing [5]. In statistical testing, one computes a probability-value (*p*-value) of a hypothesis. The hypothesis is rejected when the *p*-value is not greater than

a pre-specified level of significance, say 5%. Bootstrapping [6] is a well-known computational method for computing confidence levels when a simple mathematical formula is difficult to derive. It is a resampling method to approximate a random sample by a bootstrap sample that is created by random sampling with replacement from the original single dataset. However, it is known that the naive bootstrapping tends to give overconfidence in wrong hypotheses [7]. Thus, some advanced bootstrap methods to achieve higher accuracy have been proposed [7,8,9,10]. Among others, multiscale bootstrapping [9,10] is much more accurate. In this paper, we propose to apply the multiscale bootstrap to compute confidence levels, *i.e.*, p -values, of variable orderings estimated by LiNGAM.

2 Background

2.1 LiNGAM

In [4], a non-Gaussian variant of structural equation models, which is called LiNGAM, was proposed. Assume that observed data are generated from a process represented graphically by a directed acyclic graph, *i.e.*, DAG. Let us represent this DAG by a $m \times m$ adjacency matrix $\mathbf{B} = \{b_{ij}\}$ where every b_{ij} represents the connection strength from a variable x_j to another x_i in the DAG, *i.e.*, the *direct* causal effect of x_j on x_i . Moreover, let us denote by $k(i)$ a causal order of variables x_i in the DAG so that no later variable influences any earlier variable. Further, assume that the relations between variables are linear. Then we have

$$x_i = \sum_{k(j) < k(i)} b_{ij} x_j + e_i, \quad (1)$$

where e_i is an external influence variable. All external influences e_i are continuous random variables having *non-Gaussian* distributions with zero means and non-zero variances, and e_i are independent of each other so that there is no unobserved confounding variables [3]. In [4], the LiNGAM model (II) was shown to be identifiable without using any prior knowledge on the network structure. In [4], a discovery algorithm based on independent component analysis (ICA) [11], which is called LiNGAM algorithm, was also proposed to estimate $k(i)$ and b_{ij} .

2.2 Multiscale Bootstrap

Denote by \mathbf{x} a m -dimensional random variable vector and by $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ a random sample of size n from the distribution of \mathbf{x} . Further, define a function $f(\mathbf{X})$ so that $f(\mathbf{X})=0$ if a hypothesis is rejected and otherwise $f(\mathbf{X})=1$. Suppose that we obtain a $m \times n$ data matrix $\bar{\mathbf{X}}$ that is generated from \mathbf{x} , and the function $f(\bar{\mathbf{X}})=1$. Then, it is useful to evaluate how statistically reliable the value of $f(\bar{\mathbf{X}})=1$ is since the function could return 0 for another data matrix due to sample fluctuation. In [12], Felesenstein proposed to use bootstrapping [6] to evaluate such reliability. Let us denote by \mathbf{X}_q^* a q -th bootstrap sample of size n^* , which is created by random sampling with replacement from the columns of

X. In ordinary bootstrap, n^* is taken to be n . Then, the bootstrap probability p^{BP} is defined as a frequency that $f(\mathbf{X}^*)=1$:

$$p^{BP} = \frac{1}{Q} \sum_{q=1}^Q f(\mathbf{X}_q^*), \quad (2)$$

where Q is the number of bootstrap replications. A testing procedure was proposed that the hypothesis is rejected if p^{BP} is not greater than a significance level α ($0 < \alpha < 1$), say 0.05. However, it is known that p^{BP} is *biased* as a *p-value* [7].

In [9][10], a class of unbiased *p*-values was obtained. Let \mathbf{y}^* denote the \mathbf{y} vector corresponding to $\mathbf{X}^*=[\mathbf{x}_1^*, \dots, \mathbf{x}_{n^*}^*]$. Then the standard deviation of \mathbf{y}^* is proportional to $1/\sqrt{n^*}$, and its value relative to the case $n^*=n$ is called ‘scale’ of bootstrap resampling; this is defined by $\sigma=\sqrt{n/n^*}$. Then the bootstrap probability p^{BP} in (2) is a function of σ^2 , which is denoted by $p_{\sigma^2}^{BP}$ for clarity. The fundamental idea of the multiscale bootstrap [9][10] is to compute the bootstrap probability $p_{\sigma^2}^{BP}$ with the scale $\sigma^2=-1$, *i.e.*, the bootstrap sample size $n^*=-n$. Of course, it is impossible to set $n^*=-n$. Therefore, one first select several scales $\sigma>0$, computes the bootstrap probability for each of the corresponding bootstrap sample sizes n/σ^2 and extrapolates the bootstrap probabilities to $\sigma^2=-1$, *i.e.*, $n^*=-n$.

We now review a procedure to compute such unbiased *p*-values. Let us define a bootstrap *z*-value by

$$z_{\sigma^2} = -\Phi^{-1}(p_{\sigma^2}^{BP}), \quad (3)$$

where Φ^{-1} is the inverse of the distribution function Φ of the standard Gaussian distribution $N(0, 1)$. Further, let us call σz_{σ^2} a normalized bootstrap *z*-value. Then, consider to model the changes in σz_{σ^2} along the changing the scale σ by a model $\psi(\sigma^2|\boldsymbol{\beta})$, where $\boldsymbol{\beta}=[\beta_0, \dots, \beta_{h-1}]^T$ is a parameter vector of the model. Two model classes are proposed in [10]:

$$\psi_1^h(\sigma^2|\boldsymbol{\beta}) = \sum_{j=0}^{h-1} \beta_j \sigma^{2j}, \quad h \geq 1. \quad (4)$$

$$\psi_2^h(\sigma^2|\boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{h-2} \frac{\beta_j \sigma^{2j}}{1 + \beta_{h-1}(\sigma - 1)}, \quad 0 \leq \beta_{h-1} \leq 1, \quad h \geq 3. \quad (5)$$

The model (4) is reasonable when the boundary $\partial\mathcal{H}$ is smooth, and the model (5) is preferable when \mathcal{H} is a cone and $\partial\mathcal{H}$ is not smooth. To estimate the models, a number of sets of bootstrap replicates with different scales σ_d ($d=1, \dots, D$) are first created, and subsequently the bootstrap probability $p_{\sigma_d^2}^{BP}$ for each scale is computed. Note that the bootstrap sample sizes may be different from that of the original dataset. Then, a set of scales and normalized bootstrap *z*-values $\{\sigma_d, \sigma_d z_{\sigma_d^2}\}$ is obtained. Note that $z_{\sigma_d^2}$ is computed based on $p_{\sigma_d^2}^{BP}$ using (3). Finally, the model parameter vector $\boldsymbol{\beta}$ are estimated using the set of scales and

normalized bootstrap z -values. The maximum likelihood method is applied since $Qp_{\sigma^2}^{BP}$ follows a binomial distribution. A best model $\psi_{best}(\sigma^2|\beta)$ is selected using an information criterion AIC [13].

Then, a class of p -values using the best model $\psi_{best}(\sigma^2|\beta)$ is derived:

$$p_h^{MB} = \Phi \left\{ - \sum_{j=0}^{h-1} \frac{(-1 - \sigma_0^2)^j}{j!} \frac{\partial^j \psi_{best}(\sigma^2|\hat{\beta})}{\partial(\sigma^2)^j} \Big|_{\sigma_0^2} \right\}, \quad (6)$$

where σ_0^2 is taken to be unity. The right side of (6) is the first h terms of the Taylor series of the slope of z_{σ^2} at $1/\sigma = 1$, i.e., $\partial z_{\sigma^2}/\partial(1/\sigma)|_1$, around σ_0^2 . Further, p_1^{MB} turns out to be the naive bootstrap probability p_1^{BP} in (2). The larger h gives an unbiased p -value with asymptotic higher-order accuracy [10]. However, it also makes the maximum likelihood estimation less stable. In practice, $h=2$ or 3 is often used.

3 A Multiscale Bootstrap Procedure to Assessing Reliability of LiNGAM

We first define null hypotheses tested. We here focus on the following four types of hypotheses between x_i and x_j ($i \neq j$), although we can test hypotheses that describe the relations between more than two variables similarly:

1. H_{ij}^+ : a hypothesis that x_i is directly caused by x_j , and its connection strength is positive, i.e., $b_{ij} > 0$;
2. H_{ij}^- : a hypothesis that x_i is directly caused by x_j , and its connection strength is negative, i.e., $b_{ij} < 0$;
3. H_{ji}^+ : a hypothesis that x_j is directly caused by x_i , and its connection strength is positive, i.e., $b_{ji} > 0$;
4. H_{ji}^- : a hypothesis that x_j is directly caused by x_i , and its connection strength is negative, i.e., $b_{ji} < 0$.

We note that the signs of connections strengths are important and interesting in many applications [1] as well as the variable orderings. Further, this way of dividing the space based on the signs and orderings would make the boundaries of the regions be closer to be smooth than solely based on the orderings and help the multiscale bootstrap work better.

We now propose a new procedure to apply Multiscale Bootstrap to LiNGAM, which we call *MB-LiNGAM*:

MB-LiNGAM procedure

1. Select the scales $\sigma_1, \dots, \sigma_D$ ($D \geq 2$) so that $n_d^* = n/\sigma_d^2$ is an integer and choose the number of bootstrap replicates Q .
2. Generate Q bootstrap replicates $\mathbf{X}_{q,d}^*$ ($q=1, \dots, Q$) for each scale σ_d , i.e., each bootstrap sample size $n_d^* = n/\sigma_d^2$.

3. Perform LiNGAM algorithm to each bootstrap replicate $\mathbf{X}_{q,d}^*$ and compute the bootstrap probabilities $p_d^{BP}(H_{ij}^+)$ and $p_d^{BP}(H_{ij}^-)$ ($i \neq j$) for each scale σ_d , where $p_d^{BP}(H)$ denotes the bootstrap probability of a hypothesis H for scale σ_d .
 4. Compute the multiscale bootstrap p -values $p_h^{MB}(H_{ij}^+)$ and $p_h^{MB}(H_{ij}^-)$ ($i \neq j$) using the procedure in Section 2.2 more specifically (6), where $p_h^{MB}(H)$ denotes the multiscale bootstrap p -value of H with the order h .
-

In the simulations below, the ICA part of LiNGAM algorithm is run several times in Step 3. Each time the initial point of the optimization is randomly changed. The set of the estimates that achieves the largest value of an ICA objective function is used in the subsequent steps. It is a common practice to alleviate the effects of possible local maxima.

4 Simulations

We created two LiNGAM models with $m=6$ variables:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ b & 0 & 0 & 0 & 0 & 0 \\ b & 0 & 0 & 0 & 0 & 0 \\ b & b & 0 & 0 & 0 & 0 \\ 0 & b & 0 & b & 0 & 0 \\ b & b & b & 0 & b & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{bmatrix}, \quad (7)$$

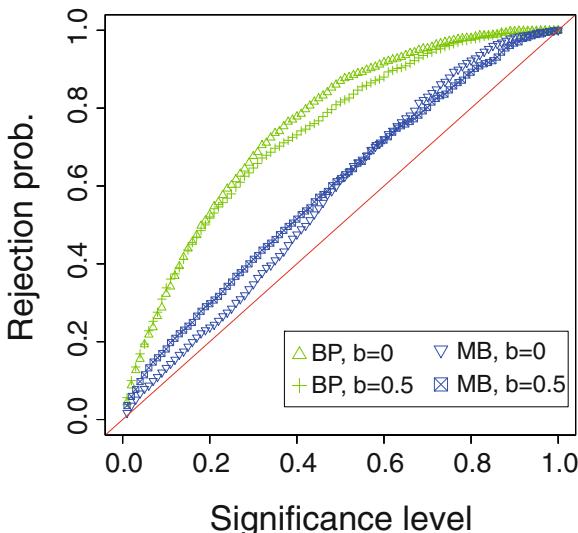


Fig. 1. Scatterplots of empirical rejection probabilities of H_{32}^+ by ordinary bootstrap and those by multiscale bootstrap versus significance levels in the six variable cases

where $b=0$ or 0.5 , and e_1 and e_2 also followed a Laplace distribution whose mean zero and variance two. We randomly generated 1280 datasets with sample size 1000 under each of the five models. Then we applied MB-LiNGAM procedure in Section 3 to the datasets. The scales σ_d were selected so that they gave integer values of bootstrap sample size and were (approximately) equally-spaced in log-scale between $1/9$ and 9 ($d=1, \dots, 13$). The number of bootstrap replicates Q was 1000, and the value h for p_h^{MB} was 3.

In Fig. 1, we also show a scatterplot of empirical rejection probabilities by ordinary bootstrap $\text{Prob}\{p^{BP}(H_{32}) < \alpha\}$ and those by multiscale bootstrap $\text{Prob}\{p_3^{MB}(H_{32}) < \alpha\}$ versus significance levels α in the six variable cases. Plots for unbiased tests should be on the diagonal. That is, their rejection probabilities should be equal to the corresponding significance levels. Most of the plots for ordinary bootstrap are far above the diagonal, indicating that ordinary bootstrap gave rather biased p -values and tended to reject reasonable hypotheses much more often than the nominal frequencies or significance levels. In contrast, the plots for multiscale bootstrap are much closer to the diagonal. This showed that multiscale bootstrap provided much better unbiased p -values.

5 Conclusion

We proposed a new procedure to evaluate statistical reliability of LiNGAM. Our procedure gives p -values of variable orderings estimated by LiNGAM and tells which orderings are more reliable.

References

1. Bollen, K.A.: Structural Equations with Latent Variables. Wiley, Chichester (1989)
2. Pearl, J.: Causality: Models, Reasoning, and Inference. Camb. Univ. Press, Cambridge (2000)
3. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search. Springer, Heidelberg (1993); 2nd edn. MIT Press, Cambridge (2000)
4. Shimizu, S., Hoyer, P.O., Hyvärinen, A., Kerminen, A.: A linear non-gaussian acyclic model for causal discovery. *J. Mach. Lear. Res.* 7, 2003–2030 (2006)
5. Lehmann, E., Romano, J.: Testing Statistical Hypotheses. Springer, Heidelberg (2008)
6. Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. Chapman and Hall, New York (1993)
7. Efron, B., Halloran, E., Holmes, S.: Bootstrap confidence levels for phylogenetic trees. *Proc. Natl. Acad. Sci. USA*, 13429–13434 (1996)
8. Hall, P.: The bootstrap and Edgeworth expansion. Springer, Heidelberg (1992)
9. Shimodaira, H.: An approximately unbiased test of phylogenetic tree selection. *Systematic Biology* 51, 492–508 (2002)
10. Shimodaira, H.: Testing regions with nonsmooth boundaries via multiscale bootstrap. *J. Statistical Planning and Inference* 138, 1227–1241 (2008)
11. Hyvärinen, A., Karhunen, J., Oja, E.: Independent component analysis. Wiley, New York (2001)
12. Felsenstein, J.: Confidence limits on phylogenies: an approach using the bootstrap. *Evolution* 39, 783–791 (1985)
13. Akaike, H.: A new look at the statistical model identification. *IEEE Trans. Automat. Control* 19(6), 716–723 (1974)

Learning with Convex Constraints

Marco Gori and Stefano Melacci

DII - University of Siena,

53100 - Siena, Italy

{marco,mela}@dii.unisi.it

Abstract. In this paper, we focus on multitask learning and discuss the notion of learning from constraints, in which they limit the space of admissible real values of the task functions. We formulate learning as a variational problem and analyze convex constraints, with special attention to the case of linear bilateral and unilateral constraints. Interestingly, we show that the solution is not always an analytic function and that it cannot be expressed by the classic kernel expansion on the training examples. We provide exact and approximate solutions and report experimental evidence of the improvement with respect to classic kernel machines.

Keywords: kernel machines; constrained optimization; regularization.

1 Introduction

The powerful framework of regularization has been playing an enormous impact in machine learning, also in the case in which more tasks are jointly involved (see e.g. [1]). Unfortunately, the curse of dimensionality, especially in presence of many tasks, makes many complex real-world problems still hard to face. A possible direction to attack those problems is to be able to express constraints on the functional space so as to restrict the hypothesis space. Following the variational framework proposed in [2], in this paper we discuss the notion of *learning from constraints*, which limits the admissible real values of the task functions. The basic idea was proposed in [6], in which the principle of stage-based learning was also advocated. We focus on convex constraints, and, in particular, we prove that the solution is still representable as kernel expansion in the special case of linear bilateral constraints, which makes it possible to the re-use kernel-like apparatus to solve the problem. Differently, in the case of unilateral constraints, even when we simply force non-negativeness of a single function, there is no classic kernel expansion to solve the problem exactly. However, we propose a technique to approximate the solution that is based on the idea, sketched in [6], of sampling the penalty term which enforces the constraint. In addition, we suggest the adoption of a *linear soft-constraining scheme* and prove that, under an appropriate choice of the regularization parameters, we can enforce the perfect satisfaction of the non-negativeness constraint, a property that holds in general for any polytope. Finally, we present an experimental report to assess the improvement of the learning from constraints scheme with respect to classic kernel machines.

2 Learning from Constraints

Given an input space X , a set of labeled samples $\mathcal{E} = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in X, \mathbf{y}_i \in \mathbb{R}^p, i = 1, \dots, \ell\}$, and a functional space \mathcal{F} , we can generalize the variational formulation given in [2] to the case of multi-task learning by choosing $\mathbf{f} = [f_1, \dots, f_p], f_j \in \mathcal{F}, j = 1, \dots, p$, such that

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \arg \min_{\mathbf{f}} \left(\sum_{k=1}^{\ell} \mathcal{L}_e(\mathbf{x}_k, \mathbf{y}_k, \mathbf{f}(\mathbf{x}_k)) d\mathbf{x} + \frac{\lambda}{2} \int_X \|P\mathbf{f}(\mathbf{x})\|^2 d\mathbf{x} \right) \\ \text{s.t. } \phi_h(\mathbf{x}) &= \phi_h^f(\mathbf{x}, f_1(\mathbf{x}), \dots, f_p(\mathbf{x})) \geq 0, h = 1, \dots, q \end{aligned} \quad (1)$$

where \mathcal{L}_e is a loss function, P is a pseudo-differential operator that is closely related to kernels [3], $\lambda > 0$ is a scalar weight, and $\phi_h, h = 1, \dots, q$, are constraints that model the relationships among $f_j, j = 1, \dots, p$. An interesting case is the one of the operator \odot_m , selected such that $\|\odot_m f\|^2 = \sum_{r=0}^m \alpha_r (d^r f(\mathbf{x}))^2$, where the $\alpha_h \geq 0$ are constant values and the derivative operator d is a scalar operator if r is even and a vector operator if r is odd. More specifically, $d^{2r} = \Delta^r = \nabla^{2r}$ and $d^{2r+1} = \nabla \nabla^{2r}$, where Δ is the Laplacian operator and ∇ is the gradient, with the additional convention $d^0 f = f$. Let us denote by P^* the adjoint of P and consider $L := P^* P$. When $p > 1$ we can construct more general differential operators with the associated L that can give rise to cross-dependencies in multi-task learning [1], but in this paper we rely on the decoupling assumption, that is the pseudo-differential operators do not produce any cross-task effect.

The solution can be given within the Lagrangian formalism [4] that requires the satisfaction of the Euler-Lagrange (EL) equations for Eq. 1. If we indicate with \mathcal{L}'_{e,f_j} the derivative of \mathcal{L}_e w.r.t. f_j , and with $\delta(\cdot)$ the Dirac delta, we have

$$\lambda L f_j + \sum_{h=1}^q \hat{\rho}_h(\mathbf{x}) \frac{\partial \phi_h^f}{\partial f_j} = - \sum_{k=1}^{\ell} \delta(\mathbf{x} - \mathbf{x}_k) \mathcal{L}'_{e,f_j}(\mathbf{x}_k, y_{k,j}, f_j(\mathbf{x}_k)) \quad (2)$$

with $j = 1, \dots, p$, that must be paired with the set of constraints $\phi_h(\mathbf{x}), h = 1, \dots, q$ and with the boundary conditions on ∂X to determine the solution. Notice that, unlike for classic function optimization, the Lagrange multipliers $\hat{\rho}_h$ for variational problems with the given subsidiary conditions are functions on X (see e.g. [4], p. 46). In general, we end up in a non-linear equation for which the classic representer theorem on which kernel machines are based does not hold. If, following the spirit of statistics and machine learning, we accept a soft-fulfillment of the constraints then the Lagrangian formulation is replaced with the optimization of an index in which we add a penalty term

$$E = \sum_{i=1}^{\ell} \sum_{j=1}^p \mathcal{L}_e(\mathbf{x}_i, y_{i,j}, f_j(\mathbf{x}_i)) + \frac{\lambda}{2} \sum_{j=1}^p \int_X \|P f_j\|^2 d\mathbf{x} + \sum_{h=1}^q \int_X \rho_h(\mathbf{x}) \mathcal{L}_c(\phi_h) d\mathbf{x} \quad (3)$$

where \mathcal{L}_e and \mathcal{L}_c are the loss functions related to the fitting of the examples and to the soft-fulfillment of the constraints, respectively, and $\rho_h(\mathbf{x})$ is an approximation of $\hat{\rho}_h(\mathbf{x})$. It can be shown that, in presence of convex loss functions and convex constraints also \mathcal{F} becomes convex, thus simplifying dramatically the problem at hand [5].

3 Learning under Linear Bilateral Constraints

Let us start considering the case of linear constraints in which $A\mathbf{f}(\mathbf{x}) = \mathbf{b}$, where $A \in \mathbb{R}^{q,p}$, $\mathbf{b} \in \mathbb{R}^q$, and $p > q$.

Lemma 1. Let $L_p = \text{diag}[P^*P]$ (the subscript on L indicates its order) and $A \in \mathbb{R}^{q,p}$ be. Then $AL_p = L_q A$.

Proof. Straightforward consequence of linearity of L_p (see [5]).

Proposition 1. Let $\det(A \cdot A') \neq 0$ be. Then the Lagrange multipliers $\hat{\rho}(\cdot)$ that yield the solution of \square are:

$$\hat{\rho}(\mathbf{x}) = -[AA']^{-1} \cdot \left(\lambda \cdot \alpha_o \cdot \mathbf{b} + \sum_{k=1}^{\ell} A\mathcal{L}'_{e,f}(\mathbf{x}_k, \mathbf{y}_k, \mathbf{f}(\mathbf{x}_k))\delta(\mathbf{x} - \mathbf{x}_k) \right). \quad (4)$$

Proof. We start from the EL equations \square in which we replace the general constraints ϕ_h with the linear constraint, and we get the proof (see [5]).

Theorem 1. Let $Q := I_p - A'[AA']^{-1}A$ be, where $I_p \in \mathbb{R}^{p,p}$ is the identity matrix. For the solution of \square under the constraints $A\mathbf{f}(\mathbf{x}) = \mathbf{b}$, the following kernel representation holds

$$\mathbf{f}(\mathbf{x}) = \hat{\psi}(\mathbf{x}) + \sum_{k=1}^{\ell} \mathbf{w}_k g(\mathbf{x} - \mathbf{x}_k), \quad \mathbf{w}_k = -\frac{Q\mathcal{L}'_{e,f}(\mathbf{x}_k, \mathbf{y}_k, \mathbf{f}(\mathbf{x}_k))}{\lambda} \quad (5)$$

where $g(\cdot)$ is the Green's function of L , $\hat{\psi}(\mathbf{x}) = \gamma_c + \psi(\mathbf{x})$, $\gamma_c := A'[AA']^{-1}\mathbf{b}$, $\psi(\cdot) \in \text{Ker}(L)$. Moreover, let $W = [\mathbf{w}_1 | \dots | \mathbf{w}_\ell] \in \mathbb{R}^{p,\ell}$ and $Y = [\mathbf{y}_1 | \dots | \mathbf{y}_\ell] \in \mathbb{R}^{p,\ell}$ be. In the case of the quadratic loss function¹ the solution can be obtained by solving $W(\lambda I_\ell + G) = Q(Y - \Psi - \gamma_c \cdot \mathbf{1}_\ell')$, where $G \in \mathbb{R}^{\ell,\ell}$ is the Gram matrix, $\mathbf{1}_\ell$ is a vector of ℓ elements equal to 1, $\Psi = [\psi(\mathbf{x}_1) | \dots | \psi(\mathbf{x}_\ell)] \in \mathbb{R}^{p,\ell}$, and $A\psi(\mathbf{x}_i) = 0$. The product by Q is not required in the case in which the examples are coherent with the constraints.

Proof. Straightforward consequence of replacing the Lagrange multipliers given by Proposition \square into the the EL-equations (see [5]).

4 Learning under Unilateral Constraints

Let us consider the case of a single function ($p = 1$) along with the single inequality constraint $f(\mathbf{x}) \geq 0$. Assuming that \mathcal{L}_e is the quadratic loss, we start by pointing out a significant difference with respect to the problem of equality linear constraints previously discussed by a simple example.

¹ We consider \mathcal{L}_e scaled by $\frac{1}{2}$ when it is the quadratic loss function.

Example 1. Let us consider a learning task in which $X = [-2, +2] \subset \mathbb{R}$ and $\mathcal{E} = \{(-1, -1), (+1, +1)\}$. We discuss the solution in the case $P = \Delta$ (see [5] for $P = \nabla$). We notice that the minimum for the loss function on the mismatch with respect to the training set requires that $f(-1) = 0$ and $f(+1) = +1$. Moreover, apart from eventual discontinuities, for any linear piece-wise function $\Delta f(x) = \partial^2 f(x)/\partial x^2 = 0$ and, therefore, $\int_X \|Pf(x)\|^2 dx = 0$. Finally, any non-negative linear piece-wise function such that $f(-1) = 0$ and $f(+1) = +1$ is a solution.

The discussion of this example indicates that the solution may not be an analytic function and that the classic representation theorem of kernel machines does not hold. We can approximate f with a kernel expansion restricted to the a finite set, such as $\{\mathbf{x}_k\}_{k=1}^\ell$. Then the problem of Eq. 11 can be solved using Lagrange multipliers with the further assumption that they are limited to training points only, i.e. $\sum_{k=1}^\ell \rho_k \delta(\mathbf{x} - \mathbf{x}_k)$ (see [5]).

PENALTY-BASED SOLUTIONS

Alternatively, we can embed the unilateral constraint in the learning process with a penalty term $\rho \int_X \mathcal{L}_c(f(\mathbf{x})) d\mathbf{x}$, where $\rho < 0$. We start considering the case of a hinge-like penalty function, that is $\mathcal{L}_c(u) = 0$ if $u \geq 0$ and $\mathcal{L}_c(u) = u$ if $u < 0$. When $f(\mathbf{x}) < 0$, the Euler-Lagrange equations become

$$\lambda Lf(\mathbf{x}) + \sum_{k=1}^\ell (f(\mathbf{x}_k) - y_k) \delta(\mathbf{x} - \mathbf{x}_k) + \rho = 0 \quad (6)$$

whereas if $f(\mathbf{x}) \geq 0$ the term ρ must be removed. Differently, in the case of a linear penalty $\mathcal{L}_c(u) = u$, which penalizes non-negative values of $f(\mathbf{x})$ but at the same time favors positive values, the Euler-Lagrange equations are the ones of Eq. 6, the representer theorem holds and we get

$$f(\mathbf{x}) = -\frac{\rho}{\alpha_o \lambda} + \sum_{k=1}^\ell w_k g(\mathbf{x} - \mathbf{x}_k), \quad \mathbf{w} = (\lambda I_\ell + G)^{-1} \left(\mathbf{y} + \frac{\rho}{\alpha_o \lambda} \mathbf{1}_\ell \right). \quad (7)$$

Lemma 2. If $\text{Ker } L = \emptyset$ then equation 6 admits a unique solution.

Proof. The proof can easily be given by contradiction [5].

In general the constraint $f(\mathbf{x}) \geq 0$ is only partially met.

Let $B := \text{diag}[\beta_1, \dots, \beta_\ell]$ be the diagonal matrix similar to G such that $G = TBT'$, where T is orthogonal. We define $y_M := \max_k \{y_k\}$, $\beta_m = \min_k \{\beta_k\}$, and $G_M := \max_{\mathbf{x} \in X} \left\{ \sum_{k=1}^\ell g(\mathbf{x} - \mathbf{x}_k) \right\}$.

Theorem 2. Let us assume that the following conditions hold

- i) $\zeta := \lambda - G_M \left\{ 1 + \|G(\lambda I + G)^{-1} \mathbf{1}_\ell\|_2 \right\} > 0$
- ii) $|\rho| \geq \frac{\alpha_o G_M y_M^2 \lambda^3 \ell}{(\lambda + \beta_m)^2 \zeta}$

then for all coordinate w_k of \mathbf{z} we have

$$w_k \geq \frac{\rho}{\alpha_o \lambda G_M}, \quad \forall \mathbf{x} \in X : f(\mathbf{x}) \geq 0.$$

Proof. See [5].

5 Experimental Results

Given $\mathbf{f}(x) = [f_1(x), f_2(x), f_3(x)]'$, consider the bilateral constraints based on

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -3 & 5 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 7 \end{bmatrix}. \quad (8)$$

We artificially generated three mono-dimensional clusters of 50 labeled points each, randomly sampling three Gaussian distributions with different means and variances. Data from the same cluster share the same label. Labels are given by a supervisor and they are supposed to be perfectly coherent with the bilateral constraints (Fig. II top row), or noisy (Fig. II bottom row). We selected a Gaussian kernel of width $\sigma = 1$, and we set $\lambda = 10^{-2}$. The functions that are learned with and without enforcing the convex constraints are reported in the first three graphs of each row of Fig. II. We also show the L_1 norm of the residual, $\|\mathbf{b} - A\mathbf{f}(x)\|_1$.

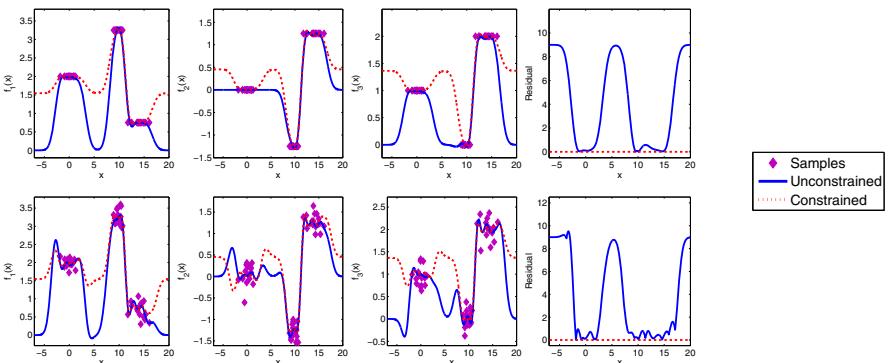


Fig. 1. $f_1(x), f_2(x), f_3(x)$ on a dataset of 150 labeled samples with and without enforcing the constraints of Eq. 8 (the L_1 norm of the residual is also shown). In the top row the labels are coherent with the constraints. In the bottom row labels are noisy.

When relying on labeled data only, the relationships of Eq. 8 are modeled only on the space regions where labeled points are distributed, and the label fitting may not be perfect due to the smoothness requirement. Differently, when constraints are introduced, the residual is zero on the entire input space.

In order to investigate the quality of the solutions proposed to enforce non-negativeness of $f(\mathbf{x})$, we selected a 2-class dataset with 1000 points (Fig. 2).

Classes are represented by blue dots and white crosses, and the corresponding targets are set to 0 and 1, respectively. Even if targets are non negative, $f(\mathbf{x})$ is not guaranteed to be strictly positive on the whole space (Fig. 2(a)). In Fig. 2(b-d), $f(\mathbf{x}) \geq 0$ is enforced by the procedures of Section 4. In Fig. 2(b) the function is constrained by the scheme based on Lagrange multipliers restricted to the training points, that assures $f(\mathbf{x}) \geq 0$ only when $\mathbf{x} \in \mathcal{E}$. Differently, Fig. 2(c) shows the result of the approach that linearly penalizes small values of the function (we set $\rho = -1.1$, $\lambda = 5$). Even if the positiveness of the function is fulfilled on the whole space, $f(\mathbf{x})$ is encouraged to assume larger values also out of the distribution of the training points. Finally, Fig. 2(d) shows a hinge loss based constraining ($\rho = 10$), that avoids penalizations of $f(\mathbf{x})$ where it is not needed.

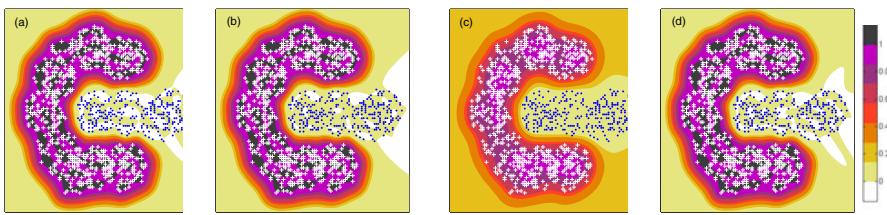


Fig. 2. A 2-class dataset (1000 samples). (a) $f(\mathbf{x})$ trained without any constraints - (b) when $f(\mathbf{x}) \geq 0$ is enforced by the Lagrange multiplier based scheme - (c) by a linear penalty - (d) by a hinge-loss penalty. $f(\mathbf{x})$ is negative on the white regions.

6 Conclusions

This paper contributes to the idea of extending the framework of learning from examples promoted by kernel machines to *learning from constraints*. Exact and approximate solutions in the case of convex functional spaces are proposed.

References

- Evgeniou, T., Micchelli, C., Pontil, M.: Learning multiple tasks with kernel methods. *Journal of Machine Learning Research* 6, 615–637 (2005)
- Poggio, T., Girosi, F.: A theory of networks for approximation and learning. Technical report, MIT (1989)
- Smola, A., Schoelkopf, B., Mueller, K.: The connection between regularization operators and support vector kernels. *Neural Networks* 11, 637–649 (1998)
- Gelfand, I., Fomin, S.: *Calculus of Variations*. Dover Publications, Inc., Mineola (1963)
- Gori, M., Melacci, S.: Learning with convex constraints. Technical report, Department of Information Engineering - University of Siena (2010)
- Gori, M.: Semantic-based regularization and Piaget's cognitive stages. *Neural Networks* 22(7), 1035–1036 (2009)

Theoretical Analysis of Cross-Validation(CV)-EM Algorithm

Takashi Takenouchi and Kazushi Ikeda

NAra Institute of Science and Technology (NAIST)
8916-5 Takayama, Ikoma, Nara, Japan
{ttakashi,kazushi}@is.naist.jp
<http://hawaii.naist.jp/~ttakashi>

Abstract. Expectation-Maximization (EM) algorithm is a typical method to estimate parameters of a model with hidden variables and is widely used for many applications. The EM algorithm is simple but sometimes overfits to specific examples and its likelihood diverges to infinite. To overcome the problem of overfitting, Shinozaki and Osterndorf have proposed the CV-EM algorithm in which the cross-validation technique is incorporated into the conventional EM algorithm, and have demonstrated validity of the algorithm with numerical experiments. In this article, we theoretically investigate properties of the CV-EM algorithm with an asymptotic analysis and reveal its mechanism of robustness.

Keywords: EM algorithm, Cross Validation, Robustness.

1 Introduction

In practical applications, a probabilistic model considering hidden variables is frequently employed to represent complicated date. For example, Boltzmann machine, mixture of Gaussians and Hidden Markov model (HMM) are typical examples of the model with hidden variables and the Expectation-Maximization (EM) algorithm [12] is a conventional method to calculate the Maximum Likelihood Estimator (MLE) of parameters of the model with hidden variables. One well recognized problem of the EM algorithm is overfitting of the model to the given training dataset. When the EM algorithm is employed, the likelihood of parameters given the training dataset arbitrary increases as complexity of the model increases, while the generalization performance for unknown dataset is not improved in general. To prevent the problem of overfitting, various approaches have been proposed. One approach is based on the model selection, which chooses an appropriate model maximizing an information criteria. The information criterion for the model with the hidden variables was developed [3] based on the framework of AIC (Akaike's information criterion) [4]. Note that the information criteria should be calculated for each individual model family and its calculation is sometimes complicated. Another approach is an employment of the Bayesian

framework: a prior distribution for the model and each parameters are introduced and its marginal likelihood is maximized with the variational method [5]. However in this approach, *a priori* design of the prior distribution is necessary.

In [6], the cross-validation technique is incorporated into the EM algorithm to overcome the problem of overfitting. One advanced point of the CV technique is that we can easily apply the technique for any model, even if calculation of the information criterion is difficult or estimation in the Bayesian framework is intractable for the model. The proposed algorithm called Cross-Validation (CV)-EM algorithm is easy to implement and effectively worked to prevent overfitting of the EM algorithm. However, the validity of CV-EM has been investigated with only numerical experiments and there was only qualitative explanation for how CV-EM prevents the overfitting for the specific data.

In this article, we theoretically investigate how CV-EM achieves the robustness for the overfitting. The rest of this article is organized as follows. In section 2, settings of problem and properties of the EM algorithm are described. The CV-EM is briefly introduced in section 3 and in section 4, we analyse property of CV-EM from viewpoint of asymptotic theory.

2 EM Algorithm

In this section, we describe basic settings and statistical properties of the EM algorithm [1]. Let \mathbf{x} be an observable random variable and z be a hidden variable, which is assumed to be in a discrete set \mathcal{Z} . We denote a probabilistic model of the variable \mathbf{x} by $p(\mathbf{x}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is a vector of parameters. Let a model of the hidden variable z and one of a complete data (\mathbf{x}, z) be $p(z|\boldsymbol{\theta})$ and $p(\mathbf{x}, z|\boldsymbol{\theta})$ respectively, which are also parametrized by $\boldsymbol{\theta}$. We assume that a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ is given and each \mathbf{x}_i is a sample of independently and identically distributed random variables, which are subject to an unknown distribution.

Purpose of the EM algorithm is to estimate parameters which maximize the log-likelihood of the parameters $\boldsymbol{\theta}$,

$$L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \log p(\mathbf{x}_i|\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \log \sum_{z \in \mathcal{Z}} p(\mathbf{x}_i, z|\boldsymbol{\theta}) \quad (1)$$

which is defined with only the observed dataset \mathcal{D} . The EM algorithm seeks the MLE by iteratively maximizing the following Q-function,

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}_t) = \frac{1}{n} \sum_{i=1}^n \sum_z p(z|\mathbf{x}_i, \boldsymbol{\theta}_t) \log p(\mathbf{x}_i, z, \boldsymbol{\theta}). \quad (2)$$

For a sequence of estimators obtained by the EM algorithm, we observe that $L(\boldsymbol{\theta}_{t+1}) \geq L(\boldsymbol{\theta}_t)$. Note that the log-likelihood is not convex with respect to $\boldsymbol{\theta}$ in general, and there is no guarantee that the EM algorithm achieves the global optimum, while the log-likelihood is non-decreasing.

While the model with hidden variables has high representation ability, overfitting of the model to the given training data is typically problematic and the

EM algorithm frequently faces the problem in the parameters estimation. For example, the Gaussian mixture model gives arbitrarily large likelihood for the training dataset if one of the Gaussian indexed by z covers only one example \mathbf{x}_i with very small variance and the other Gaussian spans the rest of the dataset. In the situation, we observe that

$$p(z|\mathbf{x}_i, \boldsymbol{\theta}) \rightarrow 1, p(\mathbf{x}_i|\boldsymbol{\theta}) \rightarrow \infty \quad (3)$$

for a specific pair (\mathbf{x}_i, z) . However, such as the model is not desirable and to overcome the problem of overfitting, various approaches has been researched. In this article, we focus on CV-EM which will be briefly introduced in the next section, and analyse theoretical aspect of CV-EM.

3 CV-EM Algorithm

The CV-EM algorithm [6] is described as follows.

1. Input: the dataset \mathcal{D} and a number of division K .
2. Divide dataset \mathcal{D} into K subset, $\{\mathcal{D}_k\}_{k=1}^K$.
3. Initialize parameters $\boldsymbol{\theta}_0^{-k}$ for each subset \mathcal{D}_k ($k = 1, \dots, K$) to $\bar{\boldsymbol{\theta}}_0$.
4. Repeat the following E-step and M-step until the parameters $\bar{\boldsymbol{\theta}}_t$ converges.
 - E-step: Calculate the posterior distribution $p(z|\mathbf{x}, \boldsymbol{\theta}_t^{-k})$ and Q-function for each sub-dataset \mathcal{D}_k ($k = 1, \dots, K$),

$$Q_k(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-k}) = \frac{1}{|\mathcal{D}_k|} \sum_{\mathbf{x}_i \in \mathcal{D}_k} \sum_{z \in \mathcal{Z}} p(z|\mathbf{x}_i, \boldsymbol{\theta}_t^{-k}) \log p(\mathbf{x}_i, z|\boldsymbol{\theta}). \quad (4)$$

- M-step: Update the parameters $\boldsymbol{\theta}_{t+1}^{-k}$ and $\bar{\boldsymbol{\theta}}_{t+1}$ as follows:

$$\boldsymbol{\theta}_{t+1}^{-k} = \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{k' \neq k} |\mathcal{D}_{k'}| Q_{k'}(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-k'}), \quad \bar{\boldsymbol{\theta}}_{t+1} = \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{k'=1}^K |\mathcal{D}_{k'}| Q_{k'}(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-k'}). \quad (5)$$

5. Output: the estimated parameters $\bar{\boldsymbol{\theta}}_{t+1}$.

In this algorithm, note that E-step and M-step for each k use different subset of dataset : in the E-step, Q-function (4) is defined with only the subset \mathcal{D}_k and in the M-step, parameters $\boldsymbol{\theta}_{t+1}^{-k}$ are updated without the subset \mathcal{D}_k as (5).

In [6], how CV-EM prevents the overfitting for the specific data point is qualitatively explained as “Probability distributions that are highly specialized to particular example cannot earn large likelihood, since the example used in the M-step does not appear in the E-step” and its validity has been investigated with only numerical experiments. In the following section, we will theoretically investigate the validity of the CV-EM algorithm.

4 Analysis of the CV-EM Algorithm

In this section, we analyse a behavior of the estimator by CV-EM with asymptotic theory and reveal how CV-EM overcomes the problem of overfitting.

4.1 Likelihood of the CV-EM Algorithm

As the case of the conventional EM algorithm, the Q-function for each k is connected to a log-likelihood $L^{-k}(\boldsymbol{\theta})$ of the parameters given sub-datasets $\{\mathcal{D}_{k'}\}_{k' \neq k}$, as

$$\begin{aligned} L^{-k}(\boldsymbol{\theta}) &= \frac{1}{n - |\mathcal{D}_k|} \sum_{\mathbf{x}_i \notin \mathcal{D}_k} \log p(\mathbf{x}_i | \boldsymbol{\theta}) \\ &= \frac{1}{n - |\mathcal{D}_k|} \left\{ \sum_{k' \neq k} |\mathcal{D}_{k'}| Q_{k'}(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-k'}) - \sum_{k' \neq k} \sum_{\mathbf{x}_i \in \mathcal{D}_{k'}} \sum_{z \in \mathcal{Z}} p(z | \mathbf{x}_i, \boldsymbol{\theta}_t^{-k'}) \log p(z | \mathbf{x}_i, \boldsymbol{\theta}) \right\}. \end{aligned}$$

From the relationship, we observe that $L^{-k}(\boldsymbol{\theta}_{t+1}^{-k}) \geq L^{-k}(\boldsymbol{\theta}_t^{-k})$. Note that for the log-likelihood given all dataset \mathcal{D} , $L(\boldsymbol{\theta}_{t+1}^{-k}) \geq L(\boldsymbol{\theta}_t^{-k})$ does not necessary hold because the parameter $\boldsymbol{\theta}_{t+1}^{-k}$ is updated without the subset \mathcal{D}_k . Also, the log-likelihood of $\bar{\boldsymbol{\theta}}_t$ given \mathcal{D} does not monotonically increase by the CV-EM.

4.2 Asymptotic Analysis for the CV-EM Algorithm

In this subsection, based on the numerical results in [6] in which larger K achieved better generalization performance, we consider n -fold cross validation ($K = n$) and assume that n is sufficiently large. In the following, we shall write $\partial_{\boldsymbol{\theta}}$ to mean $\frac{\partial}{\partial \boldsymbol{\theta}}$ and consider a relationship between estimators $\boldsymbol{\theta}_{t+1}^{-i}$ and $\bar{\boldsymbol{\theta}}_{t+1}$ satisfying the following equilibrium conditions,

$$0 = \frac{1}{n-1} \sum_{j \neq i} \partial_{\boldsymbol{\theta}} Q_j(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-j}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t+1}^{-i}}, \quad 0 = \frac{1}{n} \sum_{i=1}^n \partial_{\boldsymbol{\theta}} Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-i}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_{t+1}}, \quad (6)$$

respectively. Let us assume that $p(\mathbf{x}, z | \boldsymbol{\theta})$ is an exponential family, *i.e.*

$$p(\mathbf{x}, z | \boldsymbol{\theta}) = \exp \left(\boldsymbol{\theta}^T \mathbf{r}(\mathbf{x}, z) - k(\mathbf{x}, z) - \psi(\boldsymbol{\theta}) \right), \quad (7)$$

where \mathbf{r} is a vector of functions of (\mathbf{x}, z) , k is some function of (\mathbf{x}, z) and

$$\psi(\boldsymbol{\theta}) = \log \int_{\mathbf{x}} \sum_{z \in \mathcal{Z}} \exp \left(\boldsymbol{\theta}^T \mathbf{r}(\mathbf{x}, z) - k(\mathbf{x}, z) \right) d\mathbf{x}. \quad (8)$$

For estimators $\boldsymbol{\theta}_{t+1}^{-i}$ and $\bar{\boldsymbol{\theta}}$, the following lemma holds.

Lemma 1. *Assume that $\|\boldsymbol{\theta}_{t+1}^{-i} - \bar{\boldsymbol{\theta}}_{t+1}\|$ and $\|\boldsymbol{\theta}_{t+1}^{-i} - \bar{\boldsymbol{\theta}}_t\|$ are sufficiently small respectively and we can ignore higher order terms. Then estimators $\boldsymbol{\theta}_{t+1}^{-i}$, $\bar{\boldsymbol{\theta}}_{t+1}$ and $\bar{\boldsymbol{\theta}}_t$ are connected as*

$$\boldsymbol{\theta}_{t+1}^{-i} \simeq \bar{\boldsymbol{\theta}}_{t+1} - \frac{1}{n-1} G_{X,Z}(\bar{\boldsymbol{\theta}}_{t+1})^{-1} \partial_{\boldsymbol{\theta}} Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-i}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_{t+1}}, \quad (9)$$

$$\boldsymbol{\theta}_{t+1}^{-i} \simeq \bar{\boldsymbol{\theta}}_t + G_{X,Z}(\bar{\boldsymbol{\theta}}_t)^{-1} \frac{1}{n-1} \sum_{j \neq i} \partial_{\boldsymbol{\theta}} Q_j(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-j}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_t}. \quad (10)$$

$G_{X,Z}(\boldsymbol{\theta})$ is the Fisher Information matrix¹

$$G_{X,Z}(\boldsymbol{\theta}) = \int_{\mathbf{x}} \sum_{z \in \mathcal{Z}} p(\mathbf{x}, z | \boldsymbol{\theta}) \partial_{\boldsymbol{\theta}} \log p(\mathbf{x}, z | \boldsymbol{\theta}) \partial_{\boldsymbol{\theta}} \log p(\mathbf{x}, z | \boldsymbol{\theta})^T d\mathbf{x} = \partial_{\boldsymbol{\theta}} \partial_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}) \quad (11)$$

Proof. By considering the Taylor expansion of the equilibrium conditions (6) of $\boldsymbol{\theta}_{t+1}^{-i}$ around $\bar{\boldsymbol{\theta}}_{t+1}$, we observe the following.

$$\begin{aligned} 0 &= \sum_{j \neq i} \partial_{\boldsymbol{\theta}} Q_j(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-j}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t+1}^{-i}} \\ &\simeq \sum_{j=1}^n \partial_{\boldsymbol{\theta}} Q_j(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-j}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_{t+1}} - \partial_{\boldsymbol{\theta}} Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-i}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_{t+1}} + \sum_{j \neq i} \partial_{\boldsymbol{\theta}} \partial_{\boldsymbol{\theta}} Q_j(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-j}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_{t+1}} (\boldsymbol{\theta}_{t+1}^{-i} - \bar{\boldsymbol{\theta}}_{t+1}). \end{aligned}$$

The first term is the equilibrium condition (6) of $\bar{\boldsymbol{\theta}}_{t+1}$ and then is 0. Note that

$$\partial_{\boldsymbol{\theta}} \partial_{\boldsymbol{\theta}} Q_j(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-j}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_{t+1}} = - \sum_{z \in \mathcal{Z}} p(z | \mathbf{x}_j, \boldsymbol{\theta}_t^{-j}) \partial_{\boldsymbol{\theta}} \partial_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_{t+1}} = -G_{X,Z}(\bar{\boldsymbol{\theta}}_{t+1}), \quad (12)$$

because $p(\mathbf{x}, z | \boldsymbol{\theta})$ is the exponential family. Then we conclude (9).

Also equation (10) is derived by considering the Taylor expansion of the equilibrium condition (6) at $\bar{\boldsymbol{\theta}}_t$.

We consider how CV-EM differs from the conventional EM algorithm and prevents the overfitting. Using the lemma II, we obtain the following theorem.

Theorem 1. *The target function of CV-EM is approximated as*

$$\frac{1}{n} \sum_{i=1}^n Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{-i}) \simeq \frac{1}{n} \sum_{i=1}^n \sum_{z \in \mathcal{Z}} (p(z | \mathbf{x}_i, \bar{\boldsymbol{\theta}}_t) - w_{i,z}(\bar{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_{t-1}^{-i})) \log p(\mathbf{x}_i, z, \boldsymbol{\theta}), \quad (13)$$

where $w_{i,z}$ is a weight function defined by

$$w_{i,z}(\bar{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_{t-1}^{-i}) = \frac{1}{n-1} \partial_{\boldsymbol{\theta}} p(z | \mathbf{x}_i, \boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_t}^T G_{X,Z}(\bar{\boldsymbol{\theta}}_t)^{-1} \partial_{\boldsymbol{\theta}} Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}_{t-1}^{-i}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_t}.$$

Proof. By considering the Taylor expansion of the left hand side of (13) around $\bar{\boldsymbol{\theta}}_t$ and substituting (9) with index t into it, we immediately observe (13).

Remark 1. The weight function $w_{i,z}(\bar{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_{t-1}^{-i})$ is an inner product of two terms, $\partial_{\boldsymbol{\theta}} p(z | \mathbf{x}_i, \boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_t}$ and $G_{X,Z}(\bar{\boldsymbol{\theta}}_t)^{-1} \partial_{\boldsymbol{\theta}} Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}_{t-1}^{-i}) \Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}_t}$. The former term is gradient of $p(z | \mathbf{x}_i, \boldsymbol{\theta})$ and the latter term is one of the Q-function $Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}_{t-1}^{-i})$ with a metric matrix $G_{X,Z}(\bar{\boldsymbol{\theta}}_t)$. Note that the Q-function is associated with log-likelihood of parameters given \mathbf{x}_i . If the inner product is positive, *i.e.*, two gradients are in the same direction, the weight function is also positive and the posterior $p(z | \mathbf{x}_i, \boldsymbol{\theta}_t^{-i})$ is adjusted to decrease influence of a pair (\mathbf{x}_i, z) . In other words, the weight function prevents simultaneous increase of $p(z | \mathbf{x}, \boldsymbol{\theta})$ and Q-function for the specific pair (\mathbf{x}_i, z) . Due to the property of the weight function, CV-EM avoids the typical situation of overfitting as described in (3).

¹ The equation (11) holds because the model $p(\mathbf{x}, z | \boldsymbol{\theta})$ is the exponential family.

Also we observe the following relationship among estimators.

Corollary 1. *For estimators $\boldsymbol{\theta}_{t+1}^{-1}, \dots, \boldsymbol{\theta}_{t+1}^{-n}$ and $\bar{\boldsymbol{\theta}}_{t+1}$ defined in (5), we observe*

$$\bar{\boldsymbol{\theta}}_{t+1} \simeq \frac{1}{n} \sum_{i=1}^n \boldsymbol{\theta}_{t+1}^{-i}. \quad (14)$$

Proof. Consider a summation of (9) with respect to i and then we obtain (14) from the equilibrium condition (6) of $\bar{\boldsymbol{\theta}}_{t+1}$.

In CV-EM, the posterior probability $p(z|\mathbf{x}_i, \boldsymbol{\theta}_{t+1}^{-i})$ given i -th example is calculated in the E-step with the estimator $\boldsymbol{\theta}_{t+1}^{-i}$, estimated without i -th example in the M-step, and this mechanism prevents overfitting of the EM algorithm. However, if the example \mathbf{x}_i is outlier, the sequence of estimators except for $\boldsymbol{\theta}_{t+1}^{-i}$ is influenced by the outlier and then $\bar{\boldsymbol{\theta}}_{t+1}$ is also biased, because the sample mean is heavily influenced by the outlier. Then in the sense of the gross error sensitivity which is a measure of robustness [7], CV-EM has some room for further improvement.

5 Conclusions

In this article, we theoretically investigated statistical properties of the CV-EM algorithm, which has been proposed to overcome the problem of the overfitting. With the asymptotic analysis for behavior of the estimator by the CV-EM algorithm, we revealed how the CV-EM algorithm prevents the overfitting.

In [8], another improvement for the EM algorithm has been proposed for robust parameter estimation and its performance was numerically investigated. Theoretical analysis for the algorithm and comparison with the CV-EM algorithm will be interesting future work.

References

1. Bishop, C.: Pattern recognition and machine learning. Springer, Heidelberg (2006)
2. Ikeda, S.: Acceleration of the EM algorithm. Systems and Computers in Japan 31(2), 10–18 (2000)
3. Shimodaira, H.: A new criterion for selecting models from partially observed data. Lecture Notes In Statistics, p. 21. Springer, New York (1994)
4. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control 19(6), 716–723 (1974)
5. Attias, H.: A variational Bayesian framework for graphical models. Advances in Neural Information Processing Systems 12(1-2), 209–215 (2000)
6. Shinozaki, T., Ostendorf, M.: Cross-validation and aggregated EM training for robust parameter estimation. Computer Speech & Language 22(2), 185–195 (2008)
7. Hampel, F.R., Rousseeuw, P.J., Ronchetti, E.M., Stahel, W.A.: Robust Statistics. Wiley, New York (1986)
8. Shinozaki, T., Kawahara, T.: GMM and HMM training by aggregated EM algorithm with increased ensemble sizes for robust parameter estimation. In: Proc. ICASSP, Citeseer, pp. 4405–4408 (2008)

Application of BSP-Based Computational Cost Model to Predict Parallelization Efficiency of MLP Training Algorithm

Volodymyr Turchenko and Lucio Grandinetti

Department of Electronics, Informatics and Systems
University of Calabria, via P. Bucci 41C, 87036, Rende (CS), Italy
turchenko@deis.unical.it, lugran@unical.it

Abstract. The development of a computational cost model of parallel batch pattern back propagation training algorithm of a multilayer perceptron is presented in this paper. The model is developed using Bulk Synchronous Parallelism approach. The concrete parameters of the computational cost model are obtained. The developed model is used for the theoretical prediction of a parallelization efficiency of the algorithm. The predicted and real parallelization efficiencies are compared for different parallelization scenarios on two parallel high performance systems.

Keywords: Multilayer perceptron, Parallel training, Computational cost model, Efficiency prediction.

1 Introduction

Artificial neural networks (NNs) have excellent abilities to model difficult nonlinear systems. They represent a very good alternative to traditional methods for solving complex problems in many fields, including image processing, predictions, pattern recognition, robotics, optimization, etc [1]. However, most NN models require high computational load, especially in the training phase (up to hours and days). This is, indeed, the main obstacle to face for an efficient application of NNs in real-world problems. The use of general-purpose high performance computers, clusters and computational grids to speed up the training phase of NNs is the one of the ways to outperform this obstacle.

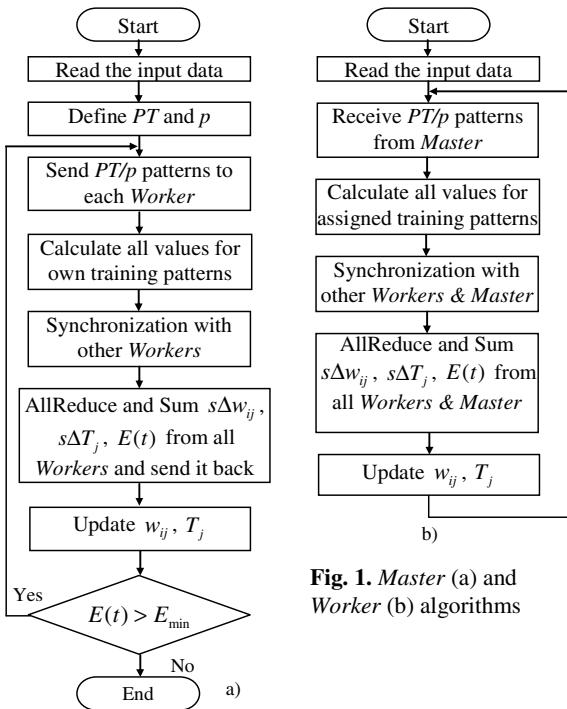
Taking into account the parallel nature of NNs, many researchers have already focused their attention on NNs parallelization on specialized computing hardware and transputers [2], [3], but these solutions are not applicable for general-purpose high performance systems. The authors of [4] investigate parallel training of multi-layer perceptron (MLP) on SMP computer, cluster and computational grid using MPI library. But their implementation of the smaller studied MLP architecture 16-10-10-1 (16 neurons in the input layer, two hidden layers with 10 neurons in each layer and one output neuron) with 270 internal connections (number of weights and thresholds) did not provide positive parallelization speedup on a cluster system due to large communication overhead, i.e. the speedup is less than 1. However, small NNs models

with the number of connections less than 270 are widely used for solving practical tasks due to better generalization abilities on the same input training data set [1]. Our MPI-implementation of the parallel batch pattern back propagation (BP) training algorithm of MLP showed good parallelization speedup on SMP computer [5], [6], [7]. One of the urgent tasks of usage of this parallel algorithm in cluster and grid environment is to develop a computational cost model of the algorithm in order to predict its parallelization efficiency on different parallel architectures available in a computational grid. It allows choosing a parallel architecture which can provide the best performance and parallelization efficiency and efficient usage of grid resources.

The goal of this paper is to develop a computational cost model of MLP parallel batch pattern BP training algorithm and to assess its accuracy on several parallel systems within a computational grid.

2 Parallel Batch Pattern BP Training Algorithm of MLP

A parallelization of an MLP with the standard sequential BP training algorithm does not provide a speedup due to high synchronization and communication overhead among parallel processors [8]. Therefore it is expedient to use the batch pattern training algorithm, which updates neurons' weights and thresholds at the end of each training epoch, i.e. after the presentation of all the input and output training patterns, instead of updating weights and thresholds after the presentation of each pattern in the



usual sequential training mode. The theoretical consideration of MLP and sequential batch pattern training algorithm are described in details in our papers [5], [6], [9].

For the development of the parallel algorithm we have used *Master - Workers* approach. The *Master* (Fig. 1) starts with the definition (i) the number of patterns PT in the training data set and (ii) the number of processors p used for the parallel executing of the training algorithm. The *Master* divides all patterns in equal parts corresponding to the number of the *Workers* and assigns one part of patterns to himself. Then the *Master* sends to

the *Workers* the numbers of the appropriate patterns to train. Each *Worker* calculates the errors of value of MLP output and hidden layers, the partial sums of delta weights $s\Delta w_{ij}$ and delta thresholds $s\Delta T_j$ and the partial SSE value for each pattern pt among the PT/p patterns assigned to him. After processing of all assigned patterns, the global operation of reduction and summation is executed. Then the summarized values of $s\Delta w_{ij}$ and $s\Delta T_j$ are sent back to all parallel processors. Using a global reducing operation and simultaneously returning the reduced values back to the *Workers* allows decreasing a communication overhead in this point. Each processor uses these values $s\Delta w_{ij}$ and $s\Delta T_j$ for updating the weights and thresholds of own copy of MLP. As the summarized value of $E(t)$ is also received as a result of the reducing operation, the *Master* decides whether to continue the training or not.

We have researched the parallelization efficiency of the algorithm for 56 growing scenarios of the problem. The expressions $S=Ts/Tp$ and $E=S/p \times 100\%$ are used to calculate a speedup and efficiency of parallelization, where Ts is the time of sequential executing the routine, Tp is the time of parallel executing of the routine on p processors of parallel system. More detailed description of the parallel algorithm and the experimental results of its parallelization are presented in [5], [6], [7].

3 Development of Computational Cost Model of the Algorithm

A computational cost model of exemplar NN training parallelism is proposed in [10] using BSP approach [11]. According to the BSP model, a computational cost of parallel program is the sum of computational costs of three sequential phases: (i) computation in each processor, using the values in its local memory, (ii) communication between parallel processors and (iii) barrier synchronization, after which communicated values become visible in the local memories of communicated processors. A computational cost of parallel program is straightforwardly calculated from a text of a routine and two architecture-specific parameters [10]: (i) g , which is a measure of the average communication cost for a fixed length message to traverse the interconnection network under continuous network load and (ii) l , which is a measure of an average time of a parallel machine to complete a barrier synchronization.

For the development of our BSP-based cost model of our algorithm we have used two MLP's structure-specific parameters, which define the computational complexity of the training algorithm: the number of patterns in the training data set N ($N=PT$ in Section 2) and the number of MLP's adjustable connections W . We have slightly modified the original BSP cost model from [10] for our algorithm as the following

$$C_{BPP} = \left[\frac{A \cdot N \cdot W}{p} \right] + l + [2 \cdot [(p-1) \cdot W] \cdot g + W]. \quad (1)$$

The detailed description of the components of the model (1) is presented in [9], [12]. However the fulfilled experimental researches showed that average absolute error of parallelization efficiency is 8% on 9 scenarios. Meanwhile several scenarios have given the difference between the real and predicted parallelization efficiencies up to 30% [9]. The results of our research are showed that the main source of an inaccuracy of the model (1) is a not correct definition of the constant A and the parameters l and g .

To fix this problem we propose to run the test routine for each possible scenario of parallelization (it executes milliseconds) on each particular parallel system which will

be used for parallel execution of this algorithm. In this case we will know exactly the real (run-time) computational rate of the algorithm and assess the real communication and barrier synchronization g and l overheads on concrete number of processors of concrete parallel system. After all considerations the new BSP-based computational cost model should be presented as the following (for one training epoch):

$$T_{WN}^p = \frac{C_{WN}}{R_W^p} + g_w^p \cdot W , \quad (2)$$

where N , W and p are the same as in the model (1), T_{WN}^p is the execution time of the MLP model with W connections and N training patterns on p processors of parallel system, C_{WN} is the computational complexity of MLP training algorithm with W connections and N training patterns, R_W^p is the real execution rate calculated at execution of test routine running one iteration of MLP model with W connections on p parallel processors, g_w^p is the real communication time of one message with W size among p parallel processors [12]. The computational complexity of MLP training algorithm is defined by the number of float point operations of multiplication/summation

$$\begin{aligned} C_{WN} = & 5 + 3LR1 + 3.5LR2 + 0.5(LR1 \times LR2) + \\ & N \times (820.5 + 3.5LR1 + 835LR2 + 10.5(LR1 \times LR2)) \end{aligned} \quad (3)$$

where $LR1$ and $LR2$ are the numbers of the input and hidden neurons respectively, $W = LR1 \times LR2 + 2 \cdot LR2 + 1$, N is the number of the training patterns. The use of the model (2) to predict the parallelization efficiency of the algorithm is presented below.

4 Experimental Researches

Our experiments were carried out on two parallel computing systems: (i) the SMP parallel supercomputer Pelikan consists of eight 64-bit dual-core processors AMD Opteron 8220 (2800 MHz), 16 GB of local RAM (667 MHz DDR2) and high-speed AMD-8131 Hyper Transport PCI-X tunnel interface (data transfer speed is 2.0 GT/s) and (ii) the computational cluster Battlecat consists of one head node (two Dual Xeon 1.6 GHz processors and 2 GB of local RAM) and 7 working nodes (dual core processor Intel Core 2 Duo 2.13 GHz and 2 GB of local RAM) connected by 1 Gbit Ethernet. We have chosen the following scenarios $W=36$ (5-5-1), $W=441$ (20-20-1) and $W=1681$ (40-40-1) for the experimental research. The time labels are provided by function *MPI_WTime()*, which measure a wall computational time of the appropriate parts of a software code. All routines are developed on C language. The Open MPI v.1.4 library is used on both systems.

The obtained results show, that the execution rate and communication time of both parallel systems are individual for each scenario. Using (3) we have calculated the computational complexity for each parallelization scenario with 200 training patterns: $C_{36}=1.3188 \times 10^7$, $C_{441}=8.7162 \times 10^8$, $C_{1681}=8.1856 \times 10^9$. Using all received figures and (2) we have calculated the predicted values of execution time, speedup and parallelization efficiency of this algorithm for each scenario (number of training epochs is fixed to 10^4) on two parallel computer systems. The real values of execution time, speedup and parallelization efficiency were obtained from a real run of this algorithm with the

same size of computational problem. The real and predicted results are grouped in Table 1 for the SMP computer and in Table 2 for the computational cluster. As it is seen from these Tables, the absolute efficiency errors are equal: average=1.4%, maximal=4.7% for the SMP computer and average=0.4%, maximal=1.5% for the computational cluster.

Table 1. Comparison of real and predicted parallelization efficiency on SMP computer

Scenario	Real/Predicted	Processors of Pelikan			
		1	2	4	8
36 (5-5-1)	Real Exec. Time,s	2.6266	1.4179	0.9110	0.8754
	Efficiency (E), %	n/a	92.5	72.1	37.5
	Predicted Exec. Time,s	2.3987	1.3306	0.7821	0.7563
	Efficiency (E), %	n/a	90.0	76.8	39.6
441 (20-20-1)	Real Exec. Time,s	23.1660	12.1323	6.6777	4.0703
	Efficiency (E), %	n/a	95.5	86.7	71.1
	Predicted Exec. Time,s	23.0936	12.0589	6.3770	4.0686
	Efficiency (E), %	n/a	96.0	90.5	70.9
1681 (40-40-1)	Real Exec. Time,s	82.5061	42.5716	22.4819	13.5305
	Efficiency (E), %	n/a	96.9	91.8	76.2
	Predicted Exec. Time,s	82.7557	42.3894	22.4783	13.2629
	Efficiency (E), %	n/a	97.5	92.0	78.0

Table 2. Comparison of real and predicted parallelization efficiency on computational cluster

Scenario	Real/Predicted	Processors of Battlecat				
		1	2	4	8	16
36 (5-5-1)	Real Exec. Time,s	2.7346	1.4231	0.8145	1.8128	2.6980
	Efficiency (E), %	n/a	96.4	84.2	18.9	6.4
	Predicted Exec. Time,s	2.7020	1.4233	0.8001	1.7227	2.6997
	Efficiency (E), %	n/a	94.9	84.4	19.6	6.2
441 (20-20-1)	Real Exec. Time,s	20.0699	10.2688	5.5063	5.4985	6.5636
	Efficiency (E), %	n/a	97.7	91.1	45.6	19.1
	Predicted Exec. Time,s	20.0813	10.2684	5.4311	5.5045	6.4589
	Efficiency (E), %	n/a	97.8	92.4	45.6	19.4
1681 (40-40-1)	Real Exec. Time,s	67.7178	34.6281	18.2120	16.2920	19.4247
	Efficiency (E), %	n/a	97.8	93.0	52.0	21.8
	Predicted Exec. Time,s	67.6431	34.6282	18.1814	16.1839	19.4250
	Efficiency (E), %	n/a	97.6	93.0	52.2	21.7

5 Conclusions

The computational cost model of the parallel batch pattern back propagation training algorithm of a multilayer perceptron is developed in this paper using Bulk Synchronous Parallelism approach. The real parameters of this model were experimentally obtained using computational complexity analysis for parallel SMP computer and computational cluster. The developed computational cost model is used for the prediction of parallelization efficiency of the parallel algorithm on both parallel systems. The real values of parallelization efficiency are obtained for the same dimensions of the parallelization problem. The comparison of the predicted and real values of parallelization efficiency is showed good prediction accuracy. The maximal absolute error of

prediction does not exceed 5% of parallelization efficiency on SMP computer and 2% on computational cluster. The developed computational cost model of the algorithm has been used for the prediction of its parallelization efficiency on different parallel systems for the development of the resource broker which provides a desired scheduling policy of minimization of the execution time of the algorithm with maximization of the parallelization efficiency in the most economic way [12].

Acknowledgment

This research is funded by the Marie Curie International Incoming Fellowship grant No. 221524 within the 7th European Community Framework Programme.

References

- [1] Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice-Hall, Englewood Cliffs (2008)
- [2] Mahapatra, S., Mahapatra, R., Chatterji, B.: A Parallel Formulation of Back-Propagation Learning on Distributed Memory Multiprocessors. *Par. Comp.* 22(12), 1661–1675 (1997)
- [3] Hanzálek, Z.: A Parallel Algorithm for Gradient Training of Feed-Forward Neural Networks. *Parallel Computing* 24(5-6), 823–839 (1998)
- [4] De Llano, R.M., Bosque, J.L.: Study of NN Training Methods in Parallel and Distributed Architectures. *Future Generation Computer Systems* 26(2), 183–190 (2010)
- [5] Turchenko, V., Grandinetti, L.: Efficiency Analysis of Parallel Batch Pattern NN Training Algorithm on General-Purpose Supercomputer. In: Omatu, S., Rocha, M.P., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., Corchado, J.M. (eds.) *IWANN 2009, Part II. LNCS*, vol. 5518, pp. 223–226. Springer, Heidelberg (2009)
- [6] Turchenko, V., Grandinetti, L.: Minimal Architecture and Training Parameters of MLP for its Efficient Parallelization. In: Proc. 5th Inter. ANNIIP Works, Italy, pp. 79–87 (2009)
- [7] Turchenko, V., Grandinetti, L.: Scalability of Enhanced Parallel Batch Pattern BP Training Algorithm on General-Purpose Supercomputers. In: De Carvalho, F., et al. (eds.) *DCAI 2010 Advances in Intelligent and Soft-Computing*. Springer, Heidelberg (in press, 2010)
- [8] Turchenko, V., Grandinetti, L.: Efficiency Research of Batch and Single Pattern MLP Parallel Training Algorithms. In: Proc. of 5th IEEE Intern. Workshop on Intelligent Data Acquis. and Advanced Computing Systems IDAACS 2009, Italy, pp. 218–224 (2009)
- [9] Turchenko, V., Grandinetti, L.: Investigation of Computational Cost Model of MLP Parallel Batch Training Algorithm. In: Proc. of 2009 IEEE Symposium on Industrial Electronics and Applications ISIEA 2009, Malaysia, pp. 983–988 (2009)
- [10] Rogers, R.O., Skillicorn, D.B.: Using the BSP cost model to optimise parallel NN training. *Future Generation Computer Systems* 14(5), 409–424 (1998)
- [11] Bisseling, R.H.: *Parallel Scientific Computation: A Structured Approach using BSP and MPI*, 334 p. Oxford University Press, Oxford (2004)
- [12] Turchenko, V., Grandinetti, L.: Strategy of Resource Brokering for Efficient Parallelization of MLP Training. In: *2010 International Conference on High Performance Computing & Simulation HPCS 2010*, France, pp. 140–149 (2010)

Dynamic Shape Learning and Forgetting*

Nikolaos Tsapanos^{1,2}, Anastasios Tefas¹, and Ioannis Pitas^{1,2}

¹ Department of Informatics, Aristotle University of Thessaloniki, Box 451, 54124, Greece

² Informatics and Telematics Institute, CERTH

{niktsap, tefas, pitas}@aiai.csd.auth.gr

Abstract. In this paper, we present a system capable of dynamically learning shapes in a way that also allows for the dynamic deletion of shapes already learned. It uses a self-balancing Binary Search Tree (BST) data structure in which we can insert shapes that we can later retrieve and also delete inserted shapes. The information concerning the inserted shapes is distributed on the tree's nodes in such a way that it is retained even after the structure of the tree changes due to insertions, deletions and rebalances these two operations can cause. Experiments show that the structure is robust enough to provide similar retrieval rates after many insertions and deletions.

1 Introduction

Object recognition by shape matching traditionally involves comparing an input test shape with various known shapes by measuring their similarity (usually by applying a Hausdorff based metric [3],[5]). The final matching returns the known shape that has yielded the maximum similarity with the input test shape. As the shape database becomes larger, however, exhaustive search becomes impractical. In order to overcome this problem, there have been a few tree structure based approaches proposed, such as the ones in [2] and [6]. An interesting property of the tree structure in [6] is that it provides means to not only learn new shapes, but to also forget previously learned shapes. In this paper, we perform various tests using this structure to determine its actual ability to learn the shape database, use the shapes it stores to classify unknown shapes and forget shapes without significant losses in classification performance for the remaining shapes. The paper is structured as follows: Section 2 briefly describes the structure, section 3 details the results of our experiments and section 4 concludes the paper.

2 Shape Trees

In this section we briefly describe the structure and the operations of the binary search tree originally presented in [6]. For our purposes, we view a shape as a set of points with 2-dimensional integer coordinates. We do so because this is the simplest way to represent a shape, though there are several other, more complicated options [7].

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant agreement No. 211471 (i3DPost).

The binary search tree consists of two types of nodes: leaf nodes and internal nodes. The shapes are stored in the leaf nodes. The internal nodes contain a template for each subtree, a matrix with the sum of the learned shapes in each subtree and they are used to traverse the tree. In order to search for a test shape in a shape tree, we must find a path of internal nodes from the root to the leaf node that corresponds to the matching shape. This can be achieved by using the internal nodes' parameters to measure the similarity of the test shape with two templates, one for each subtree. This similarity is based on the Hausdorff distance and is given by:

$$P(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} e^{-\alpha d(\mathbf{x}, \mathcal{Y})} \quad (1)$$

Where \mathcal{X} is the set of the test shape points, \mathcal{Y} is the set of the internal node template points, $d(\mathbf{x}, \mathcal{Y})$ is the distance from each point $\mathbf{x} \in \mathcal{X}$ to its closest point $\mathbf{y} \in \mathcal{Y}$ and α is a parameter determining the strictness of the similarity. Note that this similarity measure is directed and in the general case $P(\mathcal{X}, \mathcal{Y}) \neq P(\mathcal{Y}, \mathcal{X})$. The search is directed to the root of the subtree whose template yielded the highest similarity P . For practical purposes, a distance transform matrix [1] is used to calculate the measure instead of a set of points.

We will now describe how the shape tree finds the closest match of a test shape consisting of a set of points \mathcal{X} . Starting from the root of the tree we follow the path of nodes as dictated by comparing the similarities of \mathcal{X} with each of the internal node's templates until we reach a leaf. That leaf node is reported as a possible result and the search backtrack and reverses the decision on the least confident node until another leaf node is reached, which is then reported as another possible result. This can be repeated $t - 1$ times so that a list of t possible shape matches is formed. The final result is found by exhaustively searching inside this list.

The insertion operation works almost identically with regular binary search trees. A new leaf node is constructed with the input shape. The input shape is then looked up in the tree. As the search progresses, the parameters of each internal node visited are updated by adding the input shape's points into the sum matrix of the appropriate subtree. The leaf node where the search ends is replaced by a new internal node, while itself and the input shape leaf node become the new internal node's children. Finally, the reverse path to the root is followed in order to rebalance and retrain nodes in case the tree has become unbalanced after the insertion.

The deletion operation is more limited in shape trees, as only the deletion of leaf nodes is supported. Starting from the deleted node, the path to the root of the tree is followed, in order to properly update the internal nodes' parameters by subtracting the deleted shape's points from the appropriate sum matrix, and to rebalance and retrain any nodes as necessary. Since the deletion of a leaf node can leave an internal node childless and this is not allowed in shape trees, any node left childless is marked for deletion. The process is repeated until there are no more nodes marked for deletion.

Rebalancing the tree can be achieved by using the standard LL, LR, RL, RR rotations with one exception: if the top node during an LL or RR rotation has only one child. Performing an LL or RR rotation in this case would make an internal node childless. In this case, the top node is simply removed and replaced with its child. When a node is

affected by the rotation, it has to be retrained by extracting the new templates from the sum matrix of each subtree.

3 Experiments

3.1 The Database

All the experiments were conducted using the MNIST handwritten digits database [4]. This database contains 70046 28×28 images of the numbers 0 through 9 as written by 250 writers. They are separated into a subset of 60027 training images and a subset of 10019 test images.

3.2 The System

We implemented a system that uses shape trees to classify handwritten digit images. We labeled every image with the digit it depicts. The shape of each image was extracted by thresholding the brightness of the pixels to obtain a binary matrix in which the 1s formed the shape of the digit. Every shape of the training set was inserted into the tree in random order. In order to classify an input shape \mathcal{X} , we used the tree to find a list of the t closest matches $\mathcal{Y}_1 \dots \mathcal{Y}_t$, as previously explained. The final decision was the label of the shape \mathcal{Y}_f such that $f = \arg \max_i (\min(P(\mathcal{X}, \mathcal{Y}_i), P(\mathcal{Y}_i, \mathcal{X})))$. The classification was considered to be correct if the label of \mathcal{X} matched the label of \mathcal{Y}_f .

All the tests run on an AMD Athlon X2 6400 processor (each core clocked at 3.2GHz). Every averaged number is presented as *mean (standard deviation)*.

3.3 Insertion and Deletion

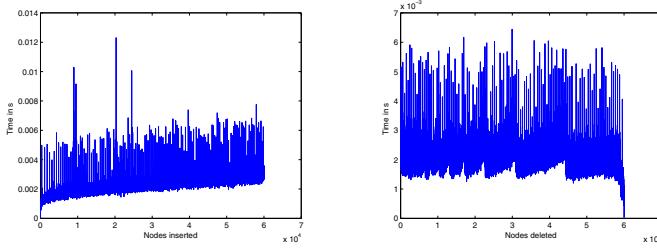
In this test we started with an empty tree and proceeded to insert all the 60027 shapes of the training set into it. We measured the time each insertion took. The graph of these values can be seen in Figure I(a). By observing the graph, we can see that it reasonably follows a logarithmic curve, as theoretically expected. The various spikes are attributed to insertions that cause more rebalances than usual. Average insertion time was 0.0026(0.0006) ms, while total tree construction time was 154 seconds.

Starting from the tree constructed in the previous test, we deleted every shape that the tree contained and measure the time each deletion took. The graph of these values can be seen in Figure I(b). Again, the spikes in the graph are attributed to some deletions causing a greater number of rebalances. Average deletion time was 0.00023(0.00029) ms, while the total time it took to empty the tree was 14 seconds.

3.4 Generalization Capabilities

Generalization capabilities were tested by first inserting the training set into the tree then using that tree to classify the test set. The number of tries was set at 64, as this number seemed to yield improved results without slowing down the classification procedure considerably.

We used the 60027 shapes for training and tested the resulting trees in the 10019 digits that were designated as the test set. The shapes are inserted into the tree in random



(a) The times (in s) that a node insertion takes vs. number of nodes already in tree.
(b) The times (in s) that a node insertion takes vs. number of nodes already deleted from the tree.

Fig. 1. Insertion and deletion times

Table 1. Average classification rates for the trees trained with 60027 shapes and tested on 10019 shapes

Digit	0	1	2	3	4	5	6	7	8	9
Rate	0.99	0.99	0.94	0.93	0.92	0.89	0.97	0.92	0.87	0.88

order. However, it is obvious that different insertion orders will produce different shape trees. Thus, in order to illustrate the effect of the insertion ordering has on the classification performance, we construct ten different trees that correspond to ten different random orderings of the inserted shapes.

The average classification rates and search times for this batch of trees are reported in Table 1. Total average classification rate was 0.9286. Results also suggest that shape trees are consistent in their performance, as there are no wild variations in classification rates.

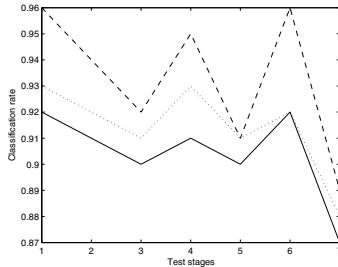
3.5 Robustness

We tested the proposed data structure's robustness to multiple insertions and deletions. We used the original training set to train the tree and the original test set to measure classification rates. Since the classes that are most likely to be confused with each other are those that correspond to the digits 3, 5 and 8, we begin by inserting these shapes into the initial tree. We denote the fact that the tree currently contains these digits as $\{3, 5, 8\}$. Thus, we consider the worst case scenario where the most difficult classes are always present inside the shape tree. We denote the insertion of additional digits with the symbol \oplus and the deletion of digits with the symbol \ominus .

After starting with the initial tree, we proceed to insert and delete digits at consecutive stages. At each new stage (after the insertions or deletions are finished), we measure the classification rates for all the digits that are into the tree at the current stage. We also monitor the classification rates for the digits 3, 5 and 8 the are present at every stage. The graph that contains the change of classification rates of the three most difficult classes over the stages of this experiment can be seen in Figure 2, while the classification rates

Table 2. Classification rates at the various stages of the robustness experiment

Digits in tree	0	1	2	3	4	5	6	7	8	9
$\{3, 5, 8\}$	-	-	-	0.92	-	0.93	-	-	0.96	-
$\{3, 5, 8\} \oplus \{0, 1\}$	0.99	0.99	-	0.91	-	0.92	-	-	0.94	-
$\{0, 1, 3, 5, 8\} \oplus \{7, 9\}$	0.99	0.99	-	0.90	-	0.91	-	0.95	0.92	0.93
$\{0, 1, 3, 5, 7, 8, 9\} \ominus \{0, 9\}$	-	0.99	-	0.91	-	0.93	-	0.98	0.95	-
$\{1, 3, 5, 7, 8\} \oplus \{4, 6\}$	-	0.99	-	0.90	0.98	0.91	0.98	0.96	0.91	-
$\{1, 3, 4, 5, 6, 7, 8\} \ominus \{1, 4, 6, 7\}$	-	-	-	0.92	-	0.92	-	-	0.96	-
$\{3, 5, 8\} \oplus \{0, 1, 2, 4, 6, 7, 9\}$	0.98	0.96	0.93	0.87	0.93	0.88	0.97	0.94	0.89	0.89

**Fig. 2.** The classification rate graphs for the digits 3 (solid line), 5 (dotted line) and 8 (dashed line)

of all the digits involved in all the stages are presented in table 2 with a '-' denoting that the tree is not trained for that specific digit and that digit is therefore not tested.

By observing Figure 2 we can see that the classifier has an easier time classifying the three digits when there are fewer overall shapes stored in the tree. The classification rate decreases when more shapes are inserted and increases as shapes are removed. Note that when the only shapes inside the tree are those that correspond to the digits 3, 5 and 8 in stage 6 of the experiment their classification rate is the same as it was in the initial tree that also contained these three digits only. Furthermore, the final classification rates match the ones in Table 1, obtained from the generalization test. This observation highlights the online training capabilities of the proposed shape tree.

3.6 Application to Human Detection

Since BSTs emphasize matching speed, shape trees can also be used in a fast human detection system. By training a tree with human silhouettes, we can scan the edge map of an image to quickly find potential matches. The resulting matches can either be thresholded, or passed through another, stronger classifier in order to be accepted as detections. Figure 3 shows a few sample detections using shape trees. These detections were carried out on multiview image data filmed at the University of Surrey under the i3dPost project. The tree was trained using silhouettes generated by the Poser software.



Fig. 3. Sample human detections using shape trees

4 Conclusion

In this paper, the performance of a shape learning structure has been tested. Experiments performed on the MNIST handwritten digit database indicate that the structure is very efficient in terms of speed, performance and scalability. It can learn new shapes with reasonable loss in classification performance and even forget previously learned while retaining its classification abilities on the data that are still stored inside it.

References

1. Borgefors, G.: Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing* 34(3), 344–371 (1986)
2. Gavrila, D.M.: A bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(8), 1408–1421 (2007)
3. Klanderman, G., Huttenlocher, D., Ruckridge, W.J.: Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Analysis and Machine Intelligence* 15(9), 850–863 (1993)
4. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(7), 2278–2324 (1998)
5. Ruckridge, W.J.: Locating objects using the Hausdorff distance. *Proceedings of Fifth International Conference on Computer Vision* 146(7), 457–464 (1995)
6. Tsapanos, N., Tefas, A., Pitas, I.: An online self-balancing binary search tree for hierarchical shape matching. In: *VISAPP*, vol. (1), pp. 591–597 (2008)
7. Zhang, D., Lu, G.: Review of shape representation and description techniques. *Pattern Recognition* 37(1), 1–19 (2004)

On-Line Ensemble-Teacher Learning through a Perceptron Rule with a Margin

Kazuyuki Hara¹, Katsuya Ono², and Seiji Miyoshi³

¹ College of Industrial Technology, Nihon University,
1-2-1, Izumi-cho, Narashino, Chiba 275-8575, Japan
hara.kazuyuki@nihon-u.ac.jp

² Tokyo Metropolitan College of Technology,
1-10-40, Higashi-oi, Shinagawa Tokyo 140-0011, Japan

³ Faculty of Engineering Science, Kansai University,
3-3-35, Yamate-cho, Suita, Osaka, 564-8680, Japan
miyoshi@ipcku.kansai-u.ac.jp

Abstract. Ensemble learning improves the performance of a learning machine by using a majority vote of many weak-learners. As an alternative, Miyoshi and Okada proposed ensemble-teacher learning. In this method, the student learns from many quasi-optimal teachers and performs better than the quasi-optimal teachers when a linear perceptron is used. When a non-linear perceptron is used, a Hebbian rule is effective; however, a perceptron rule is not effective in this case and the student cannot perform better than the quasi-optimal teachers. In this paper, we analyze ensemble-teacher learning and explain why a perceptron rule is not effective in ensemble-teacher learning. We propose a method to overcome this problem.

1 Introduction

Ensemble learning improves the performance of a learning machine by using a majority vote of many weak-learners. The majority vote is obtained by calculating the average of the weak-learnersf output. Bagging[1] or boosting[2] is a kind of ensemble learning. Ensemble learning is classified with respect to the way a new weak learner is added and the way weak learners are combined. In particular, when the weak learners are a non-linear perceptron, the space spanned by combined weak learners differs from the original space, so ensemble-learning improves the learning machine performance. [3].

Miyoshi and Okada proposed ensemble-teacher learning as an alternative method [4]. This method employs a teacher, quasi-optimal teachers, and a student. Quasi-optimal teachers learn from the teacher beforehand and are not a subject of learning. In this method, the student learns from a quasi-optimal teacher selected from a pool of many quasi-optimal teachers, and the student performs better than the quasi-optimal teachers after the learning. If a non-linear perceptron is used, a Hebbian rule is effective. However, a perceptron rule is not effective in this case and the student cannot perform better than

the quasi-optimal teachers [5]. Okada et al. showed theoretically that the student used in ensemble-teacher learning mimics an averaging mechanism of ensemble learning [6].

In this paper, we analyze ensemble-teacher learning, and derive a macroscopic learning equation. Then we show that the student learns from all the ensemble-teachers many times in microscopic time, and this is equivalent to the student learning the average of the ensemble-teachers. We also explain why the perceptron rule is not effective in ensemble-teacher learning. We propose a method to overcome this problem. The proposed method's validity is shown through computer simulation.

2 Model

In this section, we formulate an optimal teacher (latent teacher), quasi-optimal teachers (ensemble-teachers), and a student network, and the learning algorithms. We assume the latent teacher network, ensemble-teacher networks and student network receive N -dimensional input $\mathbf{x}(m) = (x_1(m), \dots, x_N(m))$ at the m -th learning iteration. We also assume that the elements $x_i(m)$ of the independently drawn input $\mathbf{x}(m)$ are uncorrelated Gaussian random variables with zero mean and $1/N$ variance; that is, the i -th element of input is drawn from a probability distribution $P(x_i)$. At the limit of $N \rightarrow \infty$, the norm of input vector $\|\mathbf{x}\|$ becomes one.

The latent teacher network is a non-linear perceptron. The ensemble-teacher networks are K non-linear perceptrons, and the student is a non-linear perceptron. The latent teacher network and the ensemble-teacher networks are not subject to training. Thus, the weight vector is fixed in the learning process. The latent teacher output is $\text{sgn}(y(m)) = \text{sgn}(\sum_{i=1}^N A_i x_i(m))$, the ensemble-teacher output is $\text{sgn}(v_k(m)) = \text{sgn}(\sum_{i=1}^N B_{ki} x_i(m))$, and the student output is $\text{sgn}(u(m)) = \text{sgn}(\sum_{i=1}^N J_i(m) x_i(m))$. Each element of the true teacher weight vector A_i , those of the ensemble-teacher weight vector B_{ki} , and those of the initial student weight vector $J_i(0)$ are drawn from a Gaussian distribution of zero mean and unit variance. Assuming the thermodynamic limit of $N \rightarrow \infty$, $\|\mathbf{A}\|$, $\|\mathbf{B}_k\|$ and $\|\mathbf{J}(0)\|$ become \sqrt{N} . At the limit, the distribution of the total input of the latent teacher $P(y)$, that of the ensemble-teacher $P(v_k)$, and that of the student $P(u)$ follow a Gaussian distribution of zero mean and unit variance. \mathbf{B}_k and \mathbf{A} are correlated with each other.

Generally, the norm of student weight vector $\|\mathbf{J}(m)\|$ changes as the time step proceeds. Therefore, the ratio l of the norm to \sqrt{N} is considered and is called the length of student weight vector \mathbf{J} . The norm at the m -th iteration is $l(m)\sqrt{N}$, and the size of $l(m)$ is $O(1)$. The distribution of the total input of the student $P(u)$ follows a Gaussian distribution of zero mean and l^2 variance in the thermodynamic limit of $N \rightarrow \infty$.

We then introduce the ensemble-teacher learning. This learning uses a true teacher, ensemble-teachers that are semi-optimal teachers, and a student. The student learns from an ensemble-teacher that is randomly selected from K

ensemble-teachers. In this paper, a perceptron rule is used for learning. The learning equation is as follows:

$$\mathbf{J}(m) = \mathbf{J}(m) + \eta \Theta(-u(m) \operatorname{sgn}(v_{k'(m)})) \operatorname{sgn}(v_{k'(m)}) \mathbf{x}(m) \quad (1)$$

Here, subscript $k'(m)$ denotes an ensemble-teacher selected at the m -th iteration.

3 Analysis of Ensemble-Teacher Learning

First, we show results of ensemble-teacher learning through a perceptron rule [5] and then we analyze the learning to show why a perceptron rule is not effective in ensemble-teacher learning.

Figure 1 shows results for ensemble-teacher learning through a perceptron rule. The horizontal axis is normalized time $t = m/N$ where m is the learning iteration. Figure 1(a) shows the time dependence of the mean error (vertical axis) between a true teacher and the student. As shown, the mean error decreased with larger K in the early stage of learning, but then became the same regardless of K . Figure 1(b) shows the percentage of ensemble-teachers that contributed to learning in every 1000 iterations. This figure shows that only 30 percent of all ensemble-teachers contributed. Therefore, the effect of learning using the majority vote obtained from many ensemble-teachers cannot be achieved.

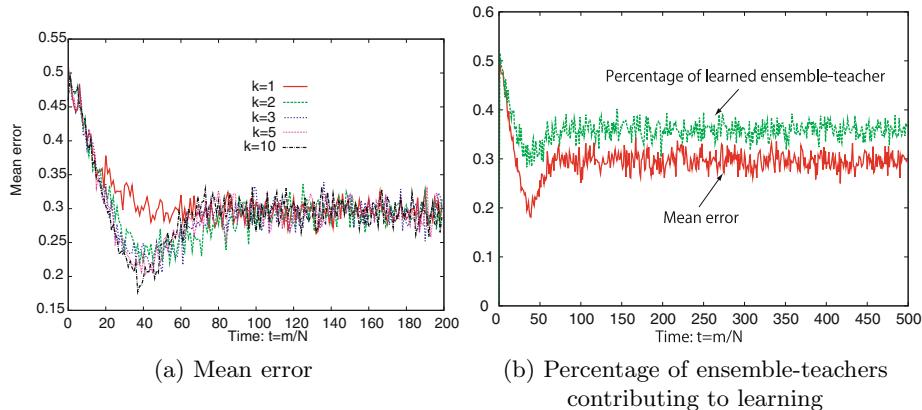


Fig. 1. Time dependence of mean error of ensemble-teacher learning when using a perceptron rule

Next, we analyze ensemble-teacher learning using a perceptron rule in the sense of statistical mechanics. For this purpose, we derive a macroscopic learning equation for the ensemble-teacher learning [5], and then discuss the effect of ensemble-teacher learning.

The microscopic learning equation at the $(m + Ndt)$ -th iteration is

$$\mathbf{J}(m + Ndt) = \mathbf{J}(m) + \eta \sum_{i=0}^{Ndt-1} \Theta(-u(m+i)v_{k'(m+i)}) \operatorname{sgn}(v_{k'(m+1)}) \mathbf{x}(m+i) \quad (2)$$

Here, $\operatorname{sgn}(x)$ outputs 1 for $x \geq 0$ and outputs -1 for $x < 0$. $\Theta(x)$ outputs 1 for $x \geq 0$ and outputs 0 for $x < 0$. Subscript $k'(m+i)$ denotes a ensemble-teacher selected at the $(m+i)$ -th iteration, and dt is microscopic time. We formulate the size of the weight vectors to be $O(\sqrt{N})$, and the norm of input \mathbf{x} is 1 in the thermodynamic limit of $N \rightarrow \infty$, so the size of $v_{k'(m+i)}$ is relatively small compared to the norm of the weight vectors. Thus, if v_j is selected as $v_{k'(m+i)}$, then $v_{k'(m+i)}$ can be replaced by the average $\langle v_j \rangle$. $u(m+i)$ can also be replaced by the average $\langle u \rangle$. An ensemble-teacher is selected by equal probability – that is, $1/K$ – and then $\sum_{i=0}^{Ndt-1} v_{k'(m+i)}$ is replaced by $\frac{Ndt}{K} \sum_{j=1}^K \langle v_j \rangle$, and the macroscopic learning equation at the $m + Ndt$ -th iteration is obtained. Thus, we get the macroscopic learning equation that shows the effect of ensemble-teacher learning.

$$\mathbf{J}(m+1) = \mathbf{J}(m) + \eta \left(\frac{1}{K} \sum_{j=1}^K \Theta(-u(m)v_j) \operatorname{sgn}(v_j) \right) \mathbf{x}(m) \quad (3)$$

From Eq. (3), it might seem that the student learns the majority vote from all of the ensemble-teacher outputs. However, Eq. (3) is the average of the ensemble outputs whose sign differs from that of the student. Consequently, as learning proceeds and the generalization error becomes smaller, the student output agrees with many ensemble-teachers and fewer ensemble-teachers are a subject of learning; consequently, the effect of the majority vote of ensemble-teachers is diminished.

4 Proposed Method

In this section, we introduce a perceptron rule with a margin [7], and then we propose a novel form of ensemble-teacher learning that can enable better performance.

As discussed in the previous section, the cause of the diminishing effect of using many ensemble-teachers is that the perceptron rule does not learn from ensemble-teachers whose output sign is the same as that of the student. To avoid this problem, we introduce a perceptron rule with a margin [7]. The learning equation is

$$\mathbf{J}_k(m+1) = \mathbf{J}_k(m) + \eta \Theta(\kappa - u(m) \operatorname{sgn}(v_k)) \operatorname{sgn}(v) \mathbf{x}(m) \quad (4)$$

Here, κ is a positive constant. As shown in Eq. (4), a perceptron rule with a margin expands the learnable region in the input space, thus changing the dynamics of ensemble-teacher learning and improving the learning ability. When

$\kappa \rightarrow \infty$, this learning rule is identical to a Hebbian rule, and when $\kappa \rightarrow 0$, it is identical to a perceptron rule. In other words, a perceptron rule with a margin is a middle way between a Hebbian rule and a perceptron rule.

We can use this learning rule in ensemble-teacher learning instead of a perceptron rule. We derive the macroscopic learning equation of the proposed method as follows:

$$\mathbf{J}(m+1) = \mathbf{J}(m) + \eta \left(\frac{1}{K} \sum_{j=1}^K \Theta(\kappa - u(m) \operatorname{sgn}(v_j)) \cdot \operatorname{sgn}(v_j) \right) \mathbf{x}(m). \quad (5)$$

Because κ is a positive constant, the student learns from an ensemble-teacher even if the student's sign agrees with that of the ensemble-teacher and the problem described in Sec. B can be eased.

5 Results

In this section, we show the validity of the proposed method through computer simulation results. For all computer simulations, the number of input elements $N = 1000$ and 1000 samples are used to calculate the mean error. Initial overlap between the true teacher and ensemble-teachers is 0.6, and that between the true teacher and the student is zero. The number of ensemble-teachers is $K = 1, 2, 3, 5$ or 10. The margin κ is 1. Figure 2(a) shows the time dependence of the mean error ε . We define the error as $\varepsilon = \Theta(-yu)$; that is, the error between the true teacher and the student. We used 1000 data to calculate the error. These results show that a perceptron rule with a margin is useful in ensemble-teacher learning compared to the original Perceptron rule from Fig. 1. Figure 2(b) shows the percentage of learned ensemble-teachers contributing in every 1000 iterations. (Results labeled "wm" are those of the proposed method and

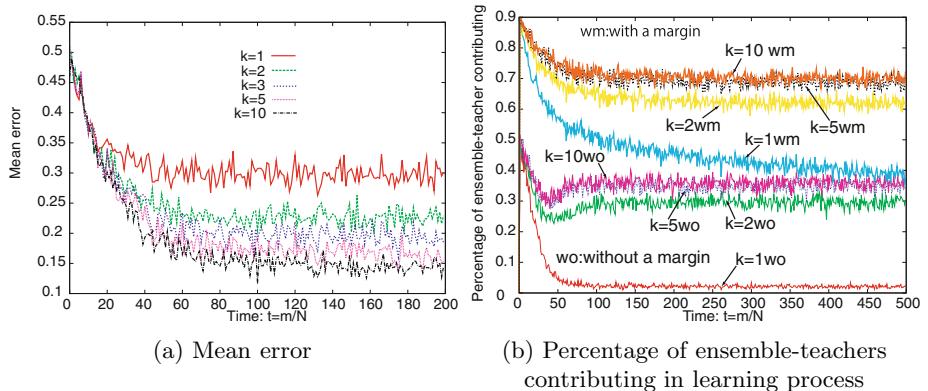


Fig. 2. Time dependence of mean error and percentage of ensemble-teachers contributing to learning (in every 1000 iterations) in ensemble-teacher learning through a perceptron rule with a margin. The margin κ is 1.

the results labeled "wo" are those of the original method.) The proposed method allows more ensemble-teachers to contribute to learning. Therefore, the validity of the proposed method is demonstrated.

6 Conclusion

In this paper, we have analyzed ensemble-teacher learning and showed that student learning from ensemble-teachers with a perceptron rule is improved by introducing a margin into the perceptron rule. The random margin for every learning iteration is also effective. Building a theoretical framework for ensemble-teacher learning will be our future work.

Acknowledgment

The authors thank Professor Masato Okada for fruitful discussions. This research was partially supported by the Ministry of Education, Culture, Sports, Science, and Technology of Japan, with a Grant-in-Aid for Scientific Research 21500228.

References

1. Breiman, L.: Bagging predictors. *Machine Learning* 24, 123 (1996)
2. Freund, Y., Shapire, R.E.: *J. Comput. Syst. Sci.* 55, 119 (1997)
3. Murata, N., Takenouchi, T., Kanamori, T., Eguchi, S.: Information Geometry of U-Boost and Bregman Divergence. *Neural Computation* 16(7), 1437–1481 (2004)
4. Miyoshi, S., Okada, M.: Statistical mechanics of online learning for ensemble-teachers. *Journal of the Physical Society of Japan* 75(4), 044002 (6 pages) (2006)
5. Utsumi, H., Miyoshi, S., Okada, M.: Statistical Mechanics of Nonlinear On-line Learning for ensemble-teachers. *J. Phys. Soc. Jpn.* 76, 114001 (2007)
6. Okada, M., Hara, K., Miyoshi, S.: Quasi-supervised learning and ensemble learning. Meeting abstracts of the Physical Society of Japan (2007) (in Japanese)
7. Hara, K., Okada, M.: On-line learning through simple perceptron with a margin. *Neural Networks* 17, 215–223 (2004)

Model of the Hippocampal Learning of Spatio-temporal Sequences

Julien Hirel, Philippe Gaussier, and Mathias Quoy

Neurocybernetic team, ETIS, CNRS - ENSEA - University of Cergy-Pontoise, F95000

Abstract. We propose a model of the hippocampus aimed at learning the timed association between subsequent sensory events. The properties of the neural network allow it to learn and predict the evolution of continuous rate-coded signals as well as the occurrence of transitory events, using both spatial and non-spatial information. The system is able to provide predictions based on the time trace of past sensory events. Performance of the neural network in the precise temporal learning of spatial and non-spatial signals is tested in a simulated experiment. The ability of the hippocampus proper to predict the occurrence of upcoming spatio-temporal events could play a crucial role in the carrying out of tasks requiring accurate time estimation and spatial localization.

1 Introduction

When an animal moves along a particular trajectory, its movement can be interpreted with two complementary perspectives: (i) as a purely spatial strategy using visual input to provide information about the current localization and adapt the behavior accordingly; (ii) as a series of actions cued to particular events, triggered by an internal time representation or path integration information. Lagarde et al. [1] showed that a robot can learn a desired trajectory both by using place-action associations or timed sequences of actions. But what about the tasks where both spatial and temporal dynamics are needed ? Is temporal and spatial information processed in different structures and somehow integrated in a larger neural network allowing spatio-temporal strategies, or is there a neural substrate directly capable of learning both spatial and temporal properties ? Some experimental tasks involve both spatial and temporal aspects [2]. In that context the animal needs time estimation, reward prediction and navigation capabilities, and a way to merge these modalities.

In this paper we propose a model where the hippocampus can learn and predict transitions between multi-modal sensory events. The model provides an internal time representation allowing the learning of timed sequences of stimuli. Using a memory of past events, we can then predict the occurrence of future transitory events or estimate the evolution of a continuous signal. Using the same architecture we are able to learn the temporal properties of various signals, spatial and/or non-spatial, in accordance to the multi-modality observed in hippocampal cells [3]. The model is tested in a simulation where a single neural network is used to learn to predict place cell activities and an unrelated non-spatial sequence of sensory stimuli.

2 Model

We previously presented a model of the hippocampus as an associative memory capable of learning the correlation between past and present sensory states and of predicting accessible states [1]. This architecture has been mainly used in two experimental contexts. First, a version using a simple memory of past states has been used in robot navigation to learn transitions between places [5]. Second, a timed memory inspired from the spectral timing model [6] was used to learn timed sequences of sensory events [7] and later for the reproduction of sequences of motor actions in a robotic experiment [8]. As previously discussed, the carrying out of certain tasks requires a strategy which integrates both temporal and spatial components, the two closely intertwined. To process all sorts of information, spatial or not, we gave our neural network the ability to predict the occurrence of punctual events as well as estimate the evolution of continuous signals over time. As an evolution over previous models, the “one-shot” learning of timings was replaced by a continuous learning. This allows the predictions to adapt their timing over time or to be forgotten if they are never fulfilled.

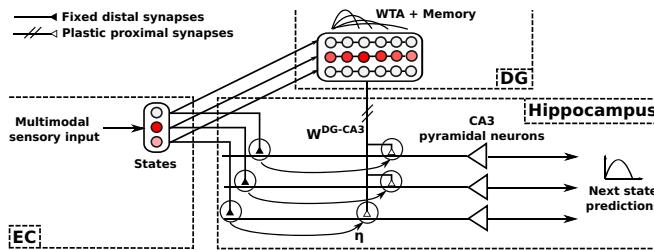


Fig. 1. Model of the state prediction neural network

In our model, EC contains the current sensory states of the robot. Multimodal signals (e.g. vision, sound, odometry etc.) are integrated. A *Winner-Take-All* (WTA) competition ensures that the activity of the most active state (i.e. current state) is transmitted to the DG memory. Each state is connected to a corresponding battery of neurons with various temporal properties in DG [7]. When the current state changes, the memory is reset and the battery corresponding to the new state is activated. This battery then gives a time trace of the delay elapsed since the state was triggered. Topological distal connections transmit state activity from EC to CA3. The activity of CA3 neurons in (II) is computed using DG-CA3 synapses only and corresponds to the prediction of future states. The learning (I) takes place in the DG-CA3 synaptic weights and is based on the Normalized Least Mean Square (NLMS) algorithm [8], with the addition of a synaptic learning modulation η .

$$x_i^{CA3}(t) = \sum_{k \in DG} W_{ik}^{DG-CA3} \cdot x_k^{DG} - \theta \quad (1)$$

$$W_{ij}^{DG-CA3}(t+dt) = W_{ij}^{DG-CA3}(t) + \alpha \cdot \eta_i(t) \cdot \frac{(x_i^{EC}(t) - x_i^{CA3}(t))}{\sum_{k \in DG} x_k^{DG}(t)^2 + \sigma_1} \cdot x_j^{DG}(t) \quad (2)$$

where θ is an activity threshold, W_{ij}^{DG-CA3} is the synaptic weight from DG neuron j to CA3 neuron i, α is the learning rate, η_i a learning modulation (see eq. 3), x_i^{EC} is the activity transmitted to neuron i by EC (i.e. the target signal for the LMS). The particularity of the NLMS over the standard LMS is the normalization term $\sum_k x_k^{DG}(t)^2 + \sigma_1$ representing the energy of the input vector. σ_1 is a small value used to avoid the divergence of the synaptic weights for very low DG values.

Each neuron of a DG battery displays a pattern of activity defined by a Gaussian function of the time elapsed since the battery was activated, with its own particular mean and variance [7]. Using the NLMS rule and a continuous signal as EC input, the neural network is capable of predicting the evolution of the signal using its internal time representation. When working with transitory input signals, like punctual events generating short bursts of neural spiking, the model learns to predict the timing of the event in relation to the time passed since the last event. The result is a bell shaped rate-coded activity, with a peak predicting the event. Since learning is performed online, the use of a LMS algorithm is problematic (the samples are not randomly distributed). As soon as a battery of DG cells is activated, the NLMS starts to learn to approximate the EC input signal and minimize the estimation error. The prediction of an event can start well before the onset of the burst of neural activity coding for this event. As a result, a predictive output signal is present while there is no activity on EC. The NLMS learning rule will converge to correct this error, thus leading to the decay of synaptic weights. Once the predicted event occurs and neural activity arises in EC, the previous decay will be balanced by the new learning of the input signal. Yet, the NLMS learning rule intrinsically leads to the same decay and learning speeds. Since transitory signals have short periods of activity (learning phase) and their prediction activity can start well in advance of their occurrence (decay phase), the previous learning of a transition is quickly forgotten. In order to learn online transitory events, the CA3 learning rule needs to be more reactive to the short period of EC activity than to the periods with no input activity. Therefore, our model includes a local modulation η of the synaptic learning rule. Its equation (3) uses the difference between the current value of the EC input signal and its mean value over a certain period of time. This results in the modulation of the synaptic learning of DG-CA3 plastic proximal connections by the variability of the activity coming from EC-CA3 distal connections. The modulation leads to a higher sensitivity of the associative learning to transitory signals: rapidly changing signals are learned faster than stable ones.

$$\eta_i(t) = |x_i^{EC}(t) - m_i^{EC}(t)| + \sigma_2 \text{ with } m_i^{EC}(t) = \gamma \cdot m_i^{EC}(t-1) + (1-\gamma) \cdot x_i^{EC}(t) \quad (3)$$

where m_i^{EC} is a sliding mean of x_i^{EC} , σ_2 is a low value setting a minimal learning modulation for CA3 and γ a parameter controlling the balance between past and current activities in the computation of the sliding mean.

Interestingly, the model is both able to approximate continuous input signals and to predict transitory events. Due to the modulation η , the synaptic weights are slowly decaying when the predicted transitions do not occur. As a side effect, the system is slower in the learning of stable continuous signals.

3 Results and Discussion

A navigation experiment where a simulated robot follows a defined trajectory was conducted (Fig. 2). Place cells are learned on a minimum activity threshold basis and their activity is computed using the azimuth of the recognized landmarks in the visual field. Each battery of DG cells is composed of 15 neurons giving a spectral decomposition of time on a period of 5 seconds.

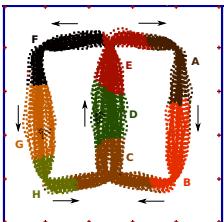


Fig. 2. 9m “ ∞ -loop” trajectories taken by the agent in the 3x3m simulated environment. 20 perfectly identifiable landmarks are placed regularly along the walls. The robot has a constant speed of 0.5m/s. One loop requires at least 18s to be completed. Time is discretized into a series of 100ms steps. The colors correspond to the most activated place cell at each location (labeled by letters).

Three types of EC signals are presented concurrently as input to CA3:

Raw place cell activity: Activity of all place cells. The DG memory gives a trace of the time passed since the arrival on the last place. The neural network learns to predict the evolution of place cell activity based on the time spent in the current place.

“Place entered” transitory signal: When the current place changes, EC triggers a short burst of activity. The DG memory is the same as for raw place cell activity. The timing of transitions from the current place to accessible neighbor places is predicted.

Non-spatial timed sequence of events: A recurring series of events, signaled by short transitory activity, forms a sequence which could represent the interaction with a person presenting various stimuli. A separate WTA competition and DG memory keeps a trace of the last event. The prediction of the timed sequence of events is learned by CA3, independently of the spatial context.

After a long period (over 50 ∞ -loops) of learning¹, we turn off synaptic modifications in order to analyze the predictions of the CA3 neurons based solely on DG activity. Figure 3 shows the EC input signal and CA3 predictions after the learning, over a period corresponding to one complete ∞ -loop. Even though the three types of signal are learned online concurrently by the same neural network,

¹ Parameters used for learning: $\alpha = 0.5$, $\sigma_1 = 0.01$, $\theta = 0.05$, $\sigma_2 = 0.01$, $\gamma = 0.5$.

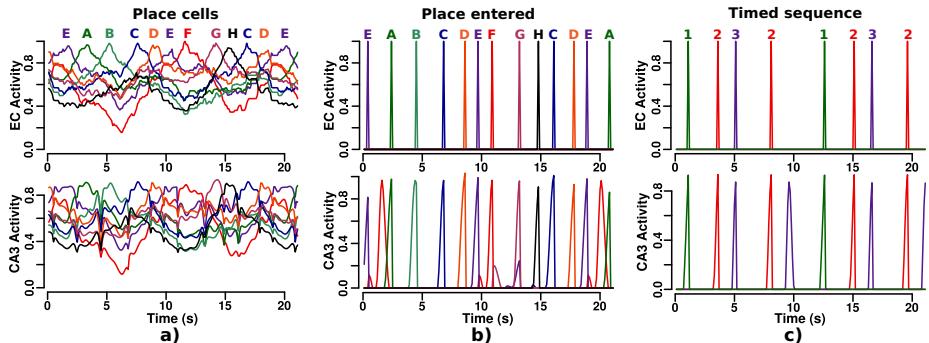


Fig. 3. **a)** Continuous prediction of all place cell activity based on the time passed since entry on the current place. **b)** Event prediction of the timing of the arrival in the next place. **c)** Prediction of the timing of the occurrence of the next non-spatial event.

they are represented separately for more clarity. We can see that the system correctly learns to predict the evolution of place cell activity, the timing of place transitions and the sequence of non-spatial events. Small drops of raw place cell prediction activity can be observed when the most active place cell changes. This is due to the reset of the DG memory and the short interval of time (100ms) needed by the first DG cells to be activated. The sum of the mean square errors (MSE) of raw place cell activity prediction, after learning and over a period of 10 ∞ -loops, is 0.035. Event predictions are learned for both spatial information (a change of the most active place cell) and non-spatial information (a repeated sequence of events, i.e. 1-2-3-2-1). All the possible events are predicted by a bell-shaped pattern of activity with the maximum value corresponding to the expected timing arrival. Predictions with a low activity can be observed for spatial events. The noise on the place cell activity can indeed lead to rare transitions between non-successive places, the low activity is a characteristic of the rareness of these occasions.

In order for the network to learn properly, the mean in (3) needs to use a short sliding window ($\gamma = 0.5$ in the simulation) so that, when the prediction of an event begins, the previous peak of activity for this event is already forgotten. This supposes local synaptic learning mechanisms for pyramidal neurons in CA3 with properties of short-term memory. A learning rate $\alpha = 0.5$ can seem high for LMS-based learning. Yet, it allows the fast learning of transitory events (which have a high η value during the short period of activity) and the slow learning of continuous signals (which have a low η value due to their low variability) needed for the stability of the LMS algorithm. Finally, an interesting property of the neural network is that, as the network learns the timing over and over, the peak of event predictions becomes more and more narrow. As fig. 3 b) and c) show, the long period of learning in the experiment leads to very sharp predictions, with a peak of activity centered on the predicted occurrence of the next event.

In conclusion, we have presented a neural network capable of learning spatial and non-spatial, continuous and transitory signals. The learned association between past and current signals results in predictive capabilities. Experimental observations of our model suggest that the learning of the timing goes through two phases: i) a short learning phase during which the amplitude of the predictions rises to reach a maximal value; ii) A long adaptation phase during which the prediction peak of activity gets progressively narrower and centered on the precise expected timing of the next event. In repetitive tasks where place transitions or sensory events have low variability in their timings, we expect to observe progressively sharper peaks of activity and reduced temporal precession for predictions. As the learning starts, it is not always clear which aspects of the task will be crucial to perform efficiently. A hypothesis is that dedicated structures learn specific components of the task (e.g. cerebellum for timed conditioning, basal ganglia for reward expectations, prefrontal cortex for movement inhibition etc.) and that superfluous information is discarded. We propose a model of the hippocampus as an associative memory extracting not only spatial but also temporal characteristics of the sensory stimuli experienced during the task. This allows to reconcile Eichenbaum's view [9] of the hippocampus as a memory and O'Keefe's view [10] of the hippocampus as a cognitive map. Finally the ability of the neural network to predict the evolution of state activity could be used in pair with the actual state activity to compute the error between the prediction and the current context. In well rehearsed tasks where the prediction should have reached a certain level of accuracy, the error signal could be used to detect anomalies or novelty. The robot would then be able to detect that the conditions of the task may have changed and adapt its behavior accordingly.

Acknowledgments. This work is supported by the CNRS, as part of a PEPS project on neuroinformatics, the DGA and the IUF. We thank JP. Banquet, B. Poucet and E. Save for useful discussions.

References

1. Lagarde, et al.: Learning new behaviors: Toward a control architecture merging spatial and temporal modalities. In: RSS (June 2008)
2. Hok, V., Lenck-Santini, P.P., Roux, S., Save, E., Muller, R.U., Poucet, B.: Goal-related activity in hippocampal place cells. *J. Neurosci.* 27(3), 472–482 (2007)
3. Wiener, S.I., Paul, C.A., Eichenbaum, H.: Spatial and behavioral correlates of hippocampal neuronal activity. *J. Neurosci.* 9(8), 2737–2763 (1989)
4. Banquet, et al.: Space-time, order and hierarchy in fronto-hippocampal system: A neural basis of personality. In: Cognitive Science Perspectives on Personality and Emotion, pp. 123–189. Elsevier Science BV, Amsterdam (1997)
5. Cuperlier, N., Quoy, M., Gaussier, P.: Neurobiologically inspired mobile robot navigation and planning. *Front Neurorobotics* 1 (2007)

6. Grossberg, S., Schmajuk, N.A.: Neural dynamics of adaptive timing temporal discrimination during associative learning. *Neural Netw.* 2(2), 79–102 (1989)
7. Moga, S., Gaussier, P., Banquet, J.P.: Sequence learning using the neural coding. In: Mira, J., Álvarez, J.R. (eds.) *IWANN 2003. LNCS*, vol. 2687, pp. 198–205. Springer, Heidelberg (2003)
8. Nagumo, J.: A learning method for system identification. *IEEE Trans. Autom. Control* 12(3), 282–287 (1967)
9. Eichenbaum, et al.: The hippocampus, memory, and place cells: is it spatial memory or a memory space? *Neuron* 23(2), 209–226 (1999)
10. O’Keefe, J., Nadel, L.: The hippocampus as a cognitive map. Oxford University Press, Oxford (1978)

Adding Nonlinear System Dynamics to Levenberg-Marquardt Algorithm for Neural Network Control

Mikel Larrea, Eloy Iglesias, and Vicente Gómez*

Department of Systems Engineering and Automatic Control
Intelligent Control Research Group

University of the Basque Country (UPV/EHU), ETSI, 48013 Bilbao
`{m.larrea,eloy.iglesias,vicente.gomez}@ehu.es`

Abstract. This paper presents a procedure to add the nonlinear system dynamics to the Levenberg-Marquardt algorithm. This algorithm is used to train a Neural Network Controller without the whole knowledge of the system to be controlled. Simulation results show a correct online training of the NN Controller.

1 Introduction

NNs are widely used as Controllers, taking advantage of their potential to adopt the behaviour of a nonlinear complex Controller. NN Controllers can be classified in two groups; “direct design method” and “indirect design method” [1]. The latter group includes the solutions where the NNs play the role of the system model, whilst the first group include the solutions where the NNs are the controllers itself.

There are many different solutions for the training of Neural Network (NN). First order methods as gradient descent, and second order algorithms as Gauss-Newton (and modifications) are used to train NNs. One of those second order algorithms is the Levenberg-Marquardt (LM) algorithm which combines a quadratic convergence rate, near a solution, with a good error decrease method, far from the solution (gradient descent). All of these algorithms do not have the ability to train a NN Controller depending on the control loop output error. To overcome this difficulty there exist many other different solutions to train NN Controllers.

This work presents an integration of the nonlinear system dynamics into the LM algorithm. This algorithm can be used to train a NN Controller, and an example of this ability will be shown.

This paper is structured as follows. It begins with a description of the control problem in Section 2, and it goes on to detail NNs and the training algorithm in Section 3 and 4. Section 5 shows some remarks on identifying models and its derivatives and the paper ends with some results in Section 6 and the conclusions in Section 7.

* This work comes under the framework of two research projects. BAIP2020 research project granted by CDTI of the Government of Spain, with permission of INGETEAM for paper publication. The Second titled ‘Aplicación de Técnicas Inteligentes en Controles Tecnológicos Avanzados’ with reference SAIOTEK08, granted by Basque Government.

2 Control Problem

The control problem to be solved is represented by an unknown nonlinear system. A general representation of this system can be expressed in the following form.

$$y(k') = M[y(k' - 1), \dots, y(k' - n), u(k' - 1), \dots, u(k' - m)] \quad (1)$$

Where $u(k')$, $y(k')$ are the system input and output respectively, and n is the nonlinear system order that must satisfy $m \leq n$. For sake of simplicity and convenience the order of the nonlinear system to be controlled is assumed to be $n = 2$. The eq 1 represents a single input single output (SISO) system.

The NN Controller training requires the knowledge of the error produced in its output to backpropagate it through the layers. The only known error in the control loop is the one produced in the output of the nonlinear system. But as the nonlinear system is partially unknown, it is necessary to identify its dynamics in order to train the Controller. The nonlinear system identification would provide the information to allow the transfer of the error produced on the system output to the NN Controller output.

In the following sections a method to overcome this problem will be presented.

3 Adding Nonlinear System Dynamic

The NN training algorithm chosen for this work is the LM algorithm applied to NNs 2. The integration of the nonlinear system dynamics into this algorithm is described in the following lines.

Considering a multilayer feedforward neural network with a hidden layer, the input output relation is represented by eq 2

$$\underline{net}_p^{m+1} = \sum_{j=1}^{S_m} W_{pj}^{m+1} \underline{a}_j^m + b_p^{m+1} \quad , \text{and} \quad \underline{a}_j^{m+1} = \underline{f}^{m+1}(\underline{net}_p^{m+1}) \quad (2)$$

where j is the input neuron number and b_p^{m+1} is the p^{th} neuron bias, \underline{f}^{m+1} is the $(m + 1)^{th}$ layer activation function vector and \underline{a}_j^{m+1} is the neuron output vector. The NN output \underline{a}_l^M is the control action (u).

The NN Controller is introduced in the control loop, where the output system error produced and the function to be minimised are represented by

$$e_i = ref_i - y_i \quad , \text{and} \quad V = \frac{1}{2} \sum_{i=1}^k e_i^T e_i \quad (3)$$

The LM algorithm calculates the update term for the weights and biases (ΔW) on the basis of the equation (eq 4).

$$\Delta W = [J^T(\underline{w}) \cdot J(\underline{w}) + \mu \cdot I]^{-1} \cdot J(\underline{w}) \cdot e(\underline{w}) \quad (4)$$

where \underline{w} is the weight vector, I is the identity matrix, μ is a variable parameter and $e(\underline{w})$ is the error vector.

$J(\underline{w})$ is the jacobian matrix that is composed by the partial derivatives ($\frac{\partial e_K(\underline{w})}{\partial w_N}$) of the errors in the NN output ($e(\underline{w})$) on weights (\underline{w}).

The jacobian matrix terms can be represented in an extended way following the chain rule.

$$\frac{\partial e_k}{\partial w_{pj}^m} = \frac{\partial e_k}{\partial \text{net}_p^{m+1}} \cdot \frac{\partial \text{net}_p^{m+1}}{\partial w_{pj}^m} \quad (5)$$

In this equation we can introduce the system dynamic without losing generality.

$$\frac{\partial e_k}{\partial w_{pj}^m} = \frac{\partial e_k}{\partial y_k} \cdot \frac{\partial y_k}{\partial u_k} \cdot \frac{\partial u_k}{\partial w_{pj}^m} \quad (6)$$

The second term ($\frac{\partial y_k}{\partial u_k}$) represents the nonlinear system dynamics. This can be re-written to consider the recursivities of the control loop and the system itself. A general representation of a nonlinear system can be expressed in the form (eq ①). Where n is the nonlinear system order that must satisfy $m \leq n$. Using this representation of a nonlinear system into eq ⑥, the following equation can be derived.

$$\frac{\partial e_k}{\partial w_{pj}^m} = \sum_{k'=1}^k \sum_{k''=0}^{k'-1} \frac{\partial e_k}{\partial y_{k'}} \cdot \frac{\partial y_{k'}}{\partial u_{k''}} \cdot \frac{\partial u_{k''}}{\partial w_{pj}^m} \quad (7)$$

The equation (eq ⑦) is made up of three terms; The first one relates to the error committed in the control loop output to the nonlinear system output, the second one relates to the nonlinear system output to the control action, and finally, the third term relates to the control action to the NN Controller weights and biases. Analyzing the eq ⑦ terms separately.

1st TERM: Deriving e_k from $y_{k'}$ the unknown term ($\frac{\partial e_k}{\partial y_{k'}}$) can be obtained. This term is usually non zero when $k' \leq k$ because of the system recursivities. Neglecting the effect on E_k of past outputs $y_{k'}$ the only non zero term is ($E_{k,k} = \frac{\partial e_k}{\partial y_k}$).

This equation represents the gradient of the error performance at instant k . If the selected performance index is mean square error (MSE), the term $E_{k,k'}$ will be as in eq ⑧.

$$E_{k,k'} = -[ref_k - y_k] \quad (8)$$

2nd TERM: Deriving $y_{k'}$ from $u_{k''}$ (eq ①) the unknown term ($\frac{\partial y_{k'}}{\partial u_{k''}}$) can be obtained. This term can in turn be decomposed in the following equation (eq ⑨) [3].

$$\frac{\partial^+ y_{k'}}{\partial u_{k''}} = \sum_{i=1}^n \frac{\partial y_{k'}}{\partial y_{k'-i}} \cdot \frac{\partial^+ y_{k'-i}}{\partial u_{k''}} + \sum_{j=1}^m \frac{\partial y_{k'}}{\partial u_{k'-j}} \cdot \frac{\partial^+ u_{k'-j}}{\partial u_{k''}} \quad (9)$$

Previous work [4][3] has shown that neglecting the terms $\frac{\partial u_{k'-j}}{\partial u_{k''}}$ when $k' - j \neq k''$, reduces the complexity of the equation while achieving good results. The second term of the equation (eq ⑨) results in the following equation (eq ⑩).

$$\frac{\partial^+ y_{k'}}{\partial u_{k''}} = \sum_{i=1}^n \frac{\partial y_{k'}}{\partial y_{k'-i}} \cdot \frac{\partial^+ y_{k'-i}}{\partial u_{k''}} + \frac{\partial y_{k'}}{\partial u_{k''}} \quad (10)$$

In this expression (eq [10]) the notation (∂^+) is used to represent partial derivatives that have implicit and explicit relationships with respect to the variables being derived.

3rd TERM:

The last term, $\frac{\partial u_{k''}}{\partial w_{pj}^m}$ is calculated on the basis of the backpropagation method.

4 Jacobian Calculation

The jacobian is a necessary element to approximate the Hessian and therefore to calculate the variation of the NN Controller weights (eq [4]).

This way, using eq [7] the jacobian $J(\underline{w})$ can be calculated. The work [2] has shown how to calculate the jacobian. The algorithm begins in the output layer with the term (Δ^M). This term is backpropagated using eq [11] through the NN.

$$\Delta^M = -\dot{F}^M(\underline{\text{net}}^M) \quad , \text{and} \quad \Delta^m = \dot{F}^m(\underline{\text{net}}^m) \cdot W^{m+1^T} \cdot \Delta^{m+1} \quad (11)$$

$$\frac{\partial \hat{V}}{\partial w_{pj}^m} = \Delta^m \cdot \underline{a}_j^{m-1} \quad , \text{and} \quad \frac{\partial \hat{V}}{\partial b_p^m} = \Delta^m \quad (12)$$

where M is the output layer and \dot{F}^m is a diagonal matrix of activation function derivatives in the m^{th} layer.

This procedure takes the error in the NN output and calculates the gradient of each weight. Those gradients are introduced in the jacobian to calculate weight modification using eq [4]. But this procedure does not consider the dynamics of the nonlinear system to be controlled. In this case, some modifications are needed.

In the case of wanting to include the nonlinear system dynamics, based on the equation (eq [5]), the term $\frac{\partial y_{k'}}{\partial u_{k''}}$ (obtained in the eq [10]) needs to be added to the backpropagation. The way that this term is considered in the LM algorithm is by integrating it in the NN output layer so it can be backpropagated (eq [13]).

$$\Delta^M = -\dot{F}^M(\underline{\text{net}}^M) \cdot \frac{\partial y_{k'}}{\partial u_{k''}} \quad (13)$$

Applying this formula and following the development presented in [2] the dynamics of the nonlinear system and the dynamics of the NN Controller are backpropagated. Therefore all the jacobian terms are calculated ($\frac{\partial e_K(\underline{w})}{\partial w_N}$) so that the weights adaptation term (ΔW) can be obtained.

Finally we want to emphasise the different meaning of the term $e(\underline{w})$ in the equation (eq [4]) used in this work. If the original work represented $e(\underline{w})$ as the error committed in the NN output, the modification carried out in this work uses $e(\underline{w}) = [(r_1 - y_1)(r_2 - y_2) \dots (r_K - y_K)]$ as the error committed in the output of the control loop.

5 Nonlinear System Identification: Use Case

In this section, a use case for the identification of an unknown nonlinear system will be presented. Continuing the development of the algorithm shown in the previous sections, a NN is used to identify the nonlinear system.

Considering a unknown nonlinear system represented by eq. [1] with order $n = 2$.

The NN Identifier used in this case has a 3-6-1 topology with sigmoid activation function on the hidden layer and a linear function on the output layer

This NN has 3 inputs; Control action u_i and system outputs y_i and y_{i-1} . Those inputs corresponds to the system derivatives $(\frac{\partial y_{k'}}{\partial y_{k'-i}}, \frac{\partial y_{k'}}{\partial u_{k''}})$ that has not been yet calculated in eq. [10].

The multilayer perceptron NNs can approximate arbitrary function and its derivatives with an accuracy fixed by the number of neurons present on the hidden layer. This property has been applied in [5] and in [6] to obtain the derivatives of the identified system $(\frac{\partial y_{k'}}{\partial y_{k'-i}}, \frac{\partial y_{k'}}{\partial u_{k''}})$.

6 Results: Nonlinear System

This is a nonlinear system proposed in [3] that has been modified from the one presented in [2] to create greater complexity.

$$y_{k+1} = 0.3 \cdot u_k^3 + \frac{2.4 \cdot y_k - 1.1 \cdot y_{k-1}}{1 + 0.8 \cdot y_k^2} \quad (14)$$

To identify the nonlinear system the NN Identifier is trained offline with random inputs inside the work range and with the following parameters: input vector length=1000 and generated by “random walk”, validation vectors length=200 , number of epochs= 1000, initial weights randomly generated within an interval calculated as in the work [8]. The identification results are presented in (Fig. 1(a)), which shows the result of the training and the two validation tests. In the latter it can be seen that the identification is satisfactory. Although there are some work ranges that have estimation errors. These errors may be due to the low representation of the work range in the training vectors.

As can be seen in Fig. 1(b), the system response shows the anticipation to the reference signal change that happens at $T = 25$. This effect is a consequence using Predictive strategy with the previous knowledge of the reference signal. The overshoot of

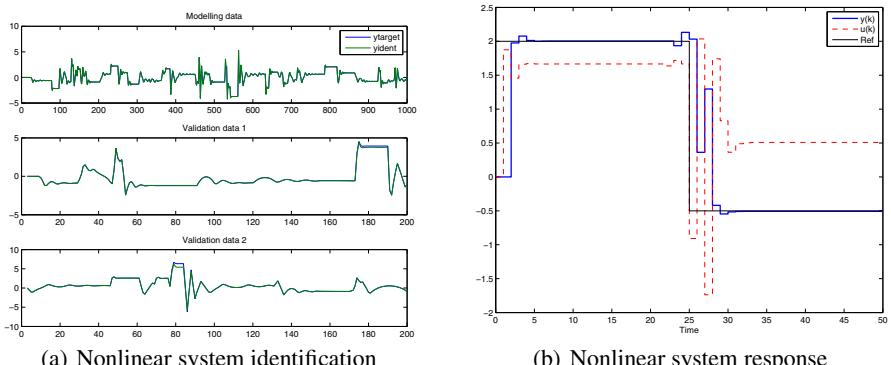


Fig. 1. Nonlinear system control

the initial step is appropriate and a penalty of abrupt changes in the control action may improve the response to the second step.

In a previous work a noisy control scheme with uncentainties in the model has been tested obtaining satisfactory results [9].

7 Conclusions

This paper introduces the integration of the dynamics of a unknown nonlinear system into the Levenberg-Marquardt algorithm for NN Controllers. Some assumptions on the nonlinear system are made to reduce the complexity of the training process.

An identification of the nonlinear system is performed to obtain the necessary terms for the NN Controller training. This identification is performed by a second NN described in detail in section 5.

The entire training process is simulated with a nonlinear system (eq.14) and the simulation results show a correct online adaptation of the NN controller and the validity of the control strategy.

References

1. Norgaard, M., Ravn, O., Poulsen, N., Hansen, L.: Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook. Springer, Heidelberg (2003)
2. Hagan, M.T., Menhaj, M.B.: Training feedforward networks with the marquardt algorithm. *IEEE Trans. on Neural Networks* 5(6), 989–993 (1994)
3. Irigoyen, E., Galván, J., Pérez-Ilzarbe, M.: Neural networks for constrained optimal control of nonlinear systems. In: *Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 4, pp. 299–304 (2000)
4. Galván, J.: Tuning of optimal neural controllers. In: *Proc. Int. Conf. on Engineering of Intelligent Systems*, pp. 213–219 (1998)
5. Fujinaka, T., Kishida, Y., Yoshioka, M., Omatsu, S.: Stabilization of double inverted pendulum with self-tuning neuro-pid. In: *Proc. of the IJCNN*, vol. 4, pp. 345–348 (2000)
6. Pirabakaran, K., Becerra, V.: Pid autotuning using neural networks and model reference adaptive control. In: *IFAC World Congress* (2002)
7. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks* 1(1), 427 (1990)
8. Irigoyen, E., Pinzolas, M.: Numerical bounds to assure initial local stability of narx multilayer perceptrons and radial basis functions. *Neurocomputing* 72(1-3), 539–547 (2008)
9. Irigoyen, E., Galván, J.B., Pérez-Ilzarbe, M.J.: A neuro predictive controller for constrained nonlinear systems. In: *IASTED Int. Conf. Artificial Intelligence and Applications*, pp. 569–574 (2003)

Some Comparisons of Model Complexity in Linear and Neural-Network Approximation

Giorgio Gnecco¹, Věra Kůrková², and Marcello Sanguineti¹

¹ Department of Communications, Computer, and System Sciences (DIST)
University of Genoa, Via Opera Pia 13, 16145 Genova, Italy
{giorgio.gnecco,marcello}@dist.unige.it

² Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, Prague 8, Czech Republic
vera@cs.cas.cz

Abstract. Capabilities of linear and neural-network models are compared from the point of view of requirements on the growth of model complexity with an increasing accuracy of approximation. Upper bounds on worst-case errors in approximation by neural networks are compared with lower bounds on these errors in linear approximation. The bounds are formulated in terms of singular numbers of certain operators induced by computational units and high-dimensional volumes of the domains of the functions to be approximated.

1 Introduction

Neural networks offer a higher flexibility in function approximation than traditional linear regression, where merely coefficients of linear combinations of fixed sets of functions are adjustable. However, certain good properties of linear models (such as uniqueness, homogeneity, and continuity of best approximation operators) are lost [1][2][3] and the optimization task in more complicated parameter sets is more difficult. Nevertheless, many successful applications suggest that the main advantage of neural networks is their reasonably moderate model complexity even in high-dimensional tasks.

Barron [4] initiated comparison of model complexity in linear and neural-network approximation by investigating rates of decrease of worst-case errors in dependence on the number of network units and the dimension of linear approximators. He found cases when upper bounds on these errors on sets of functions defined in terms of constraints on their weighted Fourier transforms are smaller than lower bounds on such errors in approximation by all linear approximators. Yet, this does not provide the desired comparison of neural networks with linear approximators, as the domains of the functions to which the bounds apply are different. Later, Kůrková and Sanguineti [5] extended Barron's comparison to various sets of functions defined on the same domain.

In this paper, we develop an approach to comparison with linear methods which can be applied to networks with a variety of computational units. Exploiting results from nonlinear approximation theory and functional analysis, we

compare upper bounds on rates of approximation by neural networks with lower bounds on worst-case errors by optimal linear approximators. We take advantage of Maurey's [6], Jones' [7], and Barron's [4] upper bounds of the form $\frac{c}{\sqrt{n}}$ on approximation by neural networks with n computational units and their recent improvement of the form $c(1 - \delta_f^2)^{n-1}$ by Kůrková and Sanguineti [8]. For sets of functions to which these bounds apply, we derive lower bounds on worst-case errors by any linear approximator. Comparing these bounds, we obtain description of cases when neural networks guarantee better accuracy of approximation with smaller model complexity than any linear approximator. Such sets depend on the type of computational units, the number d of variables of the functions to be approximated, and the volume of the d -dimensional domain where the functions are defined.

The outline of the paper follows. In Section 2, basic concepts related to linear and neural-network approximation are described. In Section 3, upper bounds on worst-case errors in neural-network approximation are given for sets of functions defined in terms of norms induced by computational units. For these sets, lower bounds on worst-case errors in linear approximation are proven in Section 4. Finally, in Section 5 estimates for neural networks and linear approximators are compared. Section 6 is a brief discussion.

2 Worst-Case Errors

One-hidden layer neural networks compute as input-output functions sets which can be formally described as

$$\text{span}_n G := \left\{ \sum_{i=1}^n w_i g_i \mid w_i \in \mathbb{R}, g_i \in G \right\},$$

where G is called a *dictionary* [9]. The approximation by the nested family $\{\text{span}_n G \mid n \in \mathbb{N}_+\}$ (where \mathbb{N}_+ denotes the set of positive integers), is sometimes called *variable-basis approximation scheme* [5]. It includes various computational models, such as perceptron neural networks, radial-basis functions, kernel models, splines with free nodes, trigonometric polynomials with variable frequencies and phases, etc. The number n of computational units can be interpreted as the *model complexity*. Typically, dictionaries are parameterized sets of functions of the form

$$G_\phi = G_\phi(Y) := \{\phi(\cdot, y) \mid y \in Y\},$$

where $\phi : \Omega \times Y \rightarrow \mathbb{R}$ is a function of two vector variables with $\Omega \subseteq \mathbb{R}^d$ representing the set of inputs and $Y \subseteq \mathbb{R}^q$ the set of parameters.

In contrast, traditional *linear* models use as approximating families nested sets

$$\text{span}\{g_1, \dots, g_n\} := \left\{ \sum_{i=1}^n w_i g_i \mid w_i \in \mathbb{R} \right\}$$

formed by linear combinations of the *first* n elements from some set $G = \{g_i \mid i \in \mathbb{N}_+\}$ with a *fixed linear ordering* (e.g., an ordered set of orthogonal polynomials).

Approximation capabilities of sets of functions can be studied in terms of *worst-case errors* formalized by the concept of *deviation*. Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be an ambient normed linear space, where the norm $\|\cdot\|_{\mathcal{X}}$ is chosen to measure approximation errors (typically, supremum norm or \mathcal{L}^2 -norm). For two subsets A and M of \mathcal{X} , the deviation of M from A is defined as

$$\delta(M, A) = \delta(M, A; \mathcal{X}) = \delta(M, A; (\mathcal{X}, \|\cdot\|_{\mathcal{X}})) := \sup_{f \in M} \inf_{g \in A} \|f - g\|_{\mathcal{X}}.$$

When the ambient space or its norm are clear from the context, we use the shorter notation.

To describe a theoretical lower bound on worst-case errors in approximation by optimal linear subspaces, Kolmogorov [10] introduced the concept of *n-width* (later called *Kolmogorov n-width*). Let \mathcal{S}_n denote the *family of all n-dimensional linear subspaces of \mathcal{X}* . The Kolmogorov *n-width* of a subset M of a normed linear space $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ is defined as the infimum of the deviations of M from all *n*-dimensional linear subspaces of \mathcal{X} , i.e.,

$$d_n(M; (\mathcal{X}, \|\cdot\|_{\mathcal{X}})) := \inf_{\mathcal{X}_n \in \mathcal{S}_n} \delta(M, \mathcal{X}_n; (\mathcal{X}, \|\cdot\|_{\mathcal{X}})) = \inf_{\mathcal{X}_n \in \mathcal{S}_n} \sup_{f \in M} \inf_{g \in \mathcal{X}_n} \|f - g\|_{\mathcal{X}}.$$

If for some subspace the infimum is achieved, then the subspace is called *optimal*. When there is no ambiguity, we write shortly $d_n(M; \mathcal{X})$ or even $d_n(M)$. Clearly, the worst-case error in linear approximation by an optimal *n*-dimensional subspace generated merely by elements of G cannot be smaller than the worst-case error in variable-basis approximation by $\text{span}_n G$. However, this does not exclude the possibility that among other linear approximators than those generated by the elements of G there exists one approximating the set M better than $\text{span}_n G$, i.e., $d_n(M) < \delta(M, \text{span}_n G)$.

Description of cases when either the opposite inequality

$$\delta(M, \text{span}_n G) < d_n(M)$$

holds or for every $f \in M$ there exists some n_0 such that for all $n \geq n_0$

$$\|f - \text{span}_n G\|_{\mathcal{X}} < d_n(M)$$

is of a great interest. For such sets M , approximation from the dictionary G outperforms any linear approximator.

3 Upper Bounds for Neural Networks

In this section, we review some upper bounds on rates of approximation by $\text{span}_n G$ which we will use for comparison with rates of linear approximation.

For any nonempty bounded subset G of a normed linear space $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, its symmetric convex closure $\text{cl}_{\mathcal{X}} \text{conv}(G \cup -G)$ uniquely determines a norm for which it forms the unit ball. It is called *G-variation* and defined for $f \in \mathcal{X}$ as

$$\|f\|_G = \|f\|_{G, \mathcal{X}} := \inf \{c > 0 \mid c^{-1} f \in \text{cl}_{\mathcal{X}} \text{conv}(G \cup -G)\} \quad (1)$$

(we write shortly $\|\cdot\|_G$ when \mathcal{X} is clear from the context). Note that G -variation can be infinite and that it is a norm on the subspace of \mathcal{X} formed by functions with finite G -variation. The general concept was introduced in [11] as an extension of variation with respect to half-spaces defined in [12].

In the next theorem, (i) is a reformulation of results by Maurey [6], Jones [7], and Barron [4] on approximation by $\text{span}_n G$ in terms of G -variation [11,13], (ii) is a straightforward corollary of (i), and (iii) is a recent result by Kůrková and Sanginetti [8] on approximation by n -fold convex combinations $\text{conv}_n G$. For any norm $\|\cdot\|$ and $r > 0$ we denote by $B_r(\|\cdot\|) = \{f \in \mathcal{X} \mid \|f\| \leq r\}$ the ball of radius r centered at the origin.

Theorem 1. *Let G be a bounded subset of a Hilbert space $(\mathcal{H}, \|\cdot\|_{\mathcal{H}})$, $s_G = \sup_{g \in G} \|g\|_{\mathcal{H}}$. Then for every positive integer n*

- (i) *for every $f \in \mathcal{H}$, $\|f - \text{span}_n G\|_{\mathcal{H}}^2 \leq \frac{s_G^2 \|f\|_{\mathcal{H}}^2 - \|f\|_{\mathcal{H}}^2}{\sqrt{n}}$;*
- (ii) *for every $r > 0$, $\delta(B_r(\|\cdot\|_G), \text{span}_n G) \leq \frac{r s_G}{\sqrt{n}}$;*
- (iii) *for every $f \in \text{conv } G$, there exists $\delta_f \in (0, 1]$ such that $\|f - \text{conv}_n G\|_{\mathcal{H}}^2 \leq (1 - \delta_f^2)^{n-1} (s_G^2 - \|f\|_{\mathcal{H}}^2)$.*

Upper bounds on approximation of functions from balls in G -variation are suitable for comparison with Kolmogorov width because for all $r > 0$

$$r d_n(G) = d_n(B_r(\|\cdot\|_G)) \quad (2)$$

as one can easily verify. To describe sets of functions contained in balls in G -variation, we take advantage of some of its estimates. For finite dictionaries $G = \{g_1, \dots, g_m\}$, it is easy to show that for a function f representable as a linear combination of elements of G , $\|f\|_G$ is the minimum of the ℓ_1 -norms of the weight vectors $w \in \mathbb{R}^m$, for which $f = \sum_{i=1}^m w_i g_i$, i.e., for every $f \in \text{span } G$, $\|f\|_G = \min\{\|w\|_1 \mid f = \sum_{i=1}^m w_i g_i, w \in \mathbb{R}^m\}$. The next theorem from [14, p. 714] shows that an analogous relationship between G -variation and \mathcal{L}^1 -norm also holds for infinite dictionaries.

Theorem 2. *Let $Y \subseteq \mathbb{R}^s$, $\Omega \subseteq \mathbb{R}^d$, μ_Y and μ_Ω be σ -finite measures on Y and Ω , resp., $\phi : \Omega \times Y \rightarrow \mathbb{R}$ such that $G_\phi(Y) = \{\phi(\cdot, y) \mid y \in Y\}$ is a bounded subset of $(\mathcal{L}_{\mu_\Omega}^p(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^p})$ with $p \in [1, \infty)$, and $w \in \mathcal{L}_{\mu_Y}^1(Y)$. Then for every $f \in \mathcal{L}_{\mu_\Omega}^p(\Omega)$ that can be represented for all $x \in \Omega$ as $f(x) = \int_Y w(y)\phi(x, y)d\mu(y)$,*

$$\|f\|_{G_\phi(Y), \mathcal{L}_{\mu_\Omega}^p(\Omega)} \leq \|w\|_{\mathcal{L}_{\mu_Y}^1(Y)}.$$

Theorem 2 provides an estimate of G_ϕ -variation under minimal assumptions needed for its formulation: G_ϕ is a bounded subset of \mathcal{X} and $w \in \mathcal{L}_{\mu_Y}^1(Y)$ (it also holds in more general functions spaces). Several results were proven earlier, under various assumptions on ϕ and Y [12,15,7,16,17,18]. Such results were used to show that balls in variations tailored to sigmoidal perceptrons and Gaussian kernel units contain large sets of smooth functions [18,19].

4 Lower Bounds on Kolmogorov n -Width

In this section, we present two methods of derivation of lower bounds on the n -widths of balls in G -variation. These bounds will be used in the next section to compare linear approximation with approximation from $\text{span}_n G$. The first method can be applied to sets of functions large enough to contain images of unit balls mapped by certain compact operators. The bound is expressed in terms of their singular numbers. Recall that a linear operator $T : \mathcal{H}_1 \rightarrow \mathcal{H}_2$ between two Hilbert spaces $(\mathcal{H}_1, \|\cdot\|_{\mathcal{H}_1})$ and $(\mathcal{H}_2, \|\cdot\|_{\mathcal{H}_2})$ is called *compact* if the image under T of every bounded set in \mathcal{H}_1 is a precompact subset of \mathcal{H}_2 (i.e., a set whose closure in the topology induced by $\|\cdot\|_{\mathcal{H}_2}$ is compact). For a compact operator T between two Hilbert spaces, the n -th *s-number* of T is defined as

$$s_n(T) = \sqrt{\lambda_n(TT^*)},$$

where T^* is the *adjoint* of T (i.e., the unique operator satisfying for every $f \in \mathcal{H}_1$ and every $g \in \mathcal{H}_2$, $\langle f, T^*g \rangle_{\mathcal{H}_1} = \langle Tf, g \rangle_{\mathcal{H}_2}$) and $\lambda_n(TT^*)$ is the n -th *eigenvalue* of the self-adjoint, non-negative, compact operator TT^* (the eigenvalues are ordered in a non-increasing sequence counting their multiplicities). If T is self-adjoint (i.e., $T = T^*$), then its singular numbers are equal to the absolute values of its eigenvalues. The following theorem [20, p. 65] states the equality between the n -width of the image of the unit ball under a compact operator T and its $(n+1)$ -th singular number.

Theorem 3. *Let $(\mathcal{H}_1, \|\cdot\|_{\mathcal{H}_1})$ and $(\mathcal{H}_2, \|\cdot\|_{\mathcal{H}_2})$ be Hilbert spaces and $T : \mathcal{H}_1 \rightarrow \mathcal{H}_2$ a compact linear operator. Then for every positive integer n ,*

$$d_n(T(B_1(\|\cdot\|_{\mathcal{H}_1})); (\mathcal{H}_2, \|\cdot\|_{\mathcal{H}_2})) = s_{n+1}(T).$$

The second method of derivation of lower bounds on n -width can be applied when balls in G -variation contain sufficiently large orthonormal subsets. It is based on the following theorem [5, p. 270].

Theorem 4. *Let A and G be subsets of a Hilbert space $(\mathcal{H}, \|\cdot\|_{\mathcal{H}})$ such that G is bounded and A is finite orthonormal with $\text{card } A = m$ and $\max_{f \in A} \|f\|_G = c_{A,G}$. Then for every positive integer $n \leq m$,*

$$d_n(G) = d_n(B_1(\|\cdot\|_G)) \geq \frac{1}{c_{A,G}} \sqrt{1 - \frac{n}{m}}.$$

We apply Theorems 3 and 4 to operators induced by computational units. Let $\Omega \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}^q$ with a measure μ_Y . For a function $\phi : \Omega \times Y \rightarrow \mathbb{R}$ and function spaces $\mathcal{F}(\Omega)$ and $\mathcal{F}(Y)$ such that the integral on the right-hand side of (3) exists for all $x \in \Omega$, an operator $T_\phi = T_{\phi, \mu_Y} : \mathcal{F}(Y) \rightarrow \mathcal{F}(\Omega)$ is defined as

$$T_\phi(w)(x) := \int_Y w(y)\phi(x, y)d\mu_Y(y). \quad (3)$$

Note that $T_\phi(w)$ can be interpreted as an *input-output function of a one-hidden layer network with infinitely many units* computing functions $\phi(\cdot, y)$, $y \in Y$ and output weights $w(y)$, $y \in Y$.

To describe properties of operators T_ϕ , we recall some definitions. A *Borel measure* on a topological space is a measure defined on the σ -algebra of its Borel subsets; it is *non-degenerate* when $\mu_\Omega(U) > 0$ for any nonempty open subset. A measure μ_Ω on Ω is σ -*finite* if there exists a countable collection of measurable sets $E_i \subseteq \Omega$ such that $\Omega = \cup_{i=1}^\infty E_i$, and $\mu_\Omega(E_i) < \infty$ for all $i \in \mathbb{N}_+$. Given two Hilbert spaces (not necessarily separable) $(\mathcal{H}_1, \|\cdot\|_{\mathcal{H}_1})$ and $(\mathcal{H}_2, \|\cdot\|_{\mathcal{H}_2})$, a bounded linear operator $T : \mathcal{H}_1 \rightarrow \mathcal{H}_2$ is called a *Hilbert-Schmidt* operator if for any orthonormal basis $\{\psi_\alpha \mid \alpha \in \mathcal{A}\}$ of \mathcal{H}_1 , $\sum_{\alpha \in \mathcal{A}} \|T\psi_\alpha\|_{\mathcal{H}_2}^2 < \infty$ [21, p. 257].

The next theorem summarizes well-known properties of the operators T_ϕ on L^2 -spaces (see [21, Proposition A.3.1, p. 257 and Proposition A.3.2, p. 259]).

Theorem 5. Let $\Omega \subseteq \mathbb{R}^d$, $Y \subseteq \mathbb{R}^q$, μ_Ω and μ_Y be non-degenerate σ -finite Borel measures on Ω and Y , resp., and $\phi \in \mathcal{L}_{\mu_\Omega \times \mu_Y}^2(\Omega \times Y)$. Then

- (i) T_ϕ maps $(\mathcal{L}_{\mu_Y}^2(Y), \|\cdot\|_{\mathcal{L}_{\mu_Y}^2})$ to $(\mathcal{L}_{\mu_\Omega}^2(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^2})$;
- (ii) T_ϕ is a Hilbert-Schmidt operator;
- (iii) T_ϕ is compact;
- (iv) if $\Omega = Y$ and ϕ is symmetric, then T_ϕ is self-adjoint, its non zero eigenvalues form a finite or countably infinite set of real numbers $\{\lambda_j\}$ satisfying $\sum_{j=1}^\infty \lambda_j^2 < \infty$, and there exists an orthonormal family $\{\psi_j\}$ in $(\mathcal{L}_{\mu_\Omega}^2(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^2})$ such that for all $g \in \mathcal{L}_{\mu_\Omega}^2(\Omega)$, $T_\phi(g) = \sum_{j=1}^N \lambda_j \langle g, \psi_j \rangle_{\mathcal{L}_{\mu_\Omega}^2} \psi_j$, where N is finite or $N = +\infty$, resp.

Applying Theorem 5 to T_ϕ , we obtain the next lower bound on the Kolmogorov n -widths of balls in G_ϕ -variation with $\phi \in \mathcal{L}_{\mu_\Omega \times \mu_Y}^2(\Omega \times Y)$, where $\mu_Y(Y)$ is finite.

Theorem 6. Let $\Omega \subseteq \mathbb{R}^d$, $Y \subseteq \mathbb{R}^q$, μ_Ω be a σ -finite non-degenerate Borel measure on Ω , μ_Y a finite non-degenerate Borel measure on Y , $\phi \in \mathcal{L}_{\mu_\Omega \times \mu_Y}^2(\Omega \times Y)$, and $G_\phi = \{\phi(\cdot, y) \mid y \in Y\}$ a bounded subset of $(\mathcal{L}_{\mu_\Omega}^2(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^2})$. Then T_ϕ is a compact operator mapping $(\mathcal{L}_{\mu_Y}^2(Y), \|\cdot\|_{\mathcal{L}_{\mu_Y}^2})$ to $(\mathcal{L}_{\mu_\Omega}^2(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^2})$ and for every positive integer n ,

$$d_n(B_1(\|\cdot\|_{G_\phi})) = d_n(G_\phi) \geq \frac{s_{n+1}(T_\phi)}{\sqrt{\mu_Y(Y)}}.$$

Proof. By Theorem 5 (i) and (iii), $T_\phi : (\mathcal{L}_{\mu_Y}^2(Y), \|\cdot\|_{\mathcal{L}_{\mu_Y}^2}) \rightarrow (\mathcal{L}_{\mu_\Omega}^2(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^2})$ is compact. Thus we can apply Theorem 3 to get $d_n(T_\phi(B_1(\|\cdot\|_{\mathcal{L}_{\mu_Y}^2}))) = s_{n+1}(T_\phi)$. As μ_Y is finite, every $w \in \mathcal{L}_{\mu_Y}^2(Y)$ is also in $\mathcal{L}_{\mu_Y}^1(Y)$ and $\|w\|_{\mathcal{L}_{\mu_Y}^1(Y)} \leq \sqrt{\mu_Y(Y)} \|w\|_{\mathcal{L}_{\mu_Y}^2(Y)}$. By Theorem 2 for $p = 2$ and $f = T_\phi(w)$, we get $\|f\|_{G_\phi} \leq \|w\|_{\mathcal{L}_{\mu_Y}^1(Y)} \leq \sqrt{\mu_Y(Y)} \|w\|_{\mathcal{L}_{\mu_Y}^2(Y)}$. Thus $B_1(\|\cdot\|_{G_\phi}) \supseteq \frac{1}{\sqrt{\mu_Y(Y)}} T_\phi(B_1(\|\cdot\|_{\mathcal{L}^2}))$ and the statement follows. \square

When ϕ is symmetric, Theorem 6 has the following corollary.

Corollary 1. Let $\Omega \subseteq \mathbb{R}^d$, μ_Ω a finite non-degenerate Borel measure on Ω , $\phi \in \mathcal{L}_{\mu_\Omega \times \mu_\Omega}^2(\Omega \times \Omega)$ be symmetric such that G_ϕ is bounded in $(\mathcal{L}_{\mu_\Omega}^2(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^2})$, and $\{\lambda_j\}$ be a sequence of eigenvalues of T_ϕ ordered non increasingly in absolute

values. Then for every positive integer n ,

$$d_n(B_1(\|\cdot\|_{G_\phi})) = d_n(G_\phi) \geq \frac{|\lambda_{n+1}|}{\sqrt{\mu_\Omega(\Omega)}}.$$

Proof. By Theorem 6, $d_n(B_1(\|\cdot\|_{G_\phi})) \geq \frac{s_{n+1}(T_\phi)}{\sqrt{\mu_\Omega(\Omega)}}$. As ϕ is symmetric, T_ϕ is self-adjoint, which implies that $s_{n+1}(T_\phi) = |\lambda_{n+1}|$. So the statement follows. \square

Using as a lower bound on the Kolmogorov n -width the estimate from Theorem 4 instead of the one from Theorem 3, we obtain the next theorem.

Theorem 7. Let $\Omega \subseteq \mathbb{R}^d$, μ_Ω be a finite non-degenerate Borel measure on Ω , $\phi \in \mathcal{L}_{\mu_\Omega \times \mu_\Omega}^2(\Omega \times \Omega)$ be symmetric such that G_ϕ is bounded in $\mathcal{L}_{\mu_\Omega}^2(\Omega)$, $\{\lambda_j\}$ be a sequence of eigenvalues of $T_\phi : \mathcal{L}_{\mu_\Omega}^2(\Omega) \rightarrow \mathcal{L}_{\mu_\Omega}^2(\Omega)$ ordered non increasingly in absolute values, and $\{\psi_j\}$ an orthonormal family of corresponding eigenfunctions. Then for all positive integers n, m such that $n \leq m$ and for $c_{m,G_\phi} = \max_{j=1,\dots,m} \|\psi_j\|_{G_\phi}$,

$$d_n(B_1(\|\cdot\|_{G_\phi})) = d_n(G_\phi) \geq \frac{1}{c_{m,G_\phi}} \sqrt{1 - \frac{n}{m}} \geq \frac{|\lambda_m|}{\sqrt{\mu_\Omega(\Omega)}} \sqrt{1 - \frac{n}{m}}.$$

Proof. Setting $A = \{\psi_1, \dots, \psi_m\}$ we obtain by Theorem 4, $d_n(B_1(\|\cdot\|_{G_\phi})) = d_n(G_\phi) \geq \frac{1}{c_{m,G_\phi}} \sqrt{1 - \frac{n}{m}}$. As $\{\psi_j\}$ are eigenfunctions of T_ϕ for all j , $\lambda_j \psi_j(x) = \int_\Omega \psi_j(y) \phi(x, y) d\mu_\Omega(y)$. By the Cauchy-Schwartz inequality $\|\psi_j\|_{\mathcal{L}_{\mu_\Omega}^1} \leq \sqrt{\mu_\Omega(\Omega)} \|\psi_j\|_{\mathcal{L}_{\mu_\Omega}^2}$, hence $\psi_j \in \mathcal{L}_{\mu_\Omega}^1(\Omega)$. As G_ϕ is bounded, the assumptions of Theorem 2 are satisfied and so $\|\psi_j\|_{G_\phi} \leq \frac{1}{|\lambda_j|} \|\psi_j\|_{\mathcal{L}_{\mu_\Omega}^1} \leq \frac{\sqrt{\mu_\Omega(\Omega)}}{|\lambda_j|} \|\psi_j\|_{\mathcal{L}_{\mu_\Omega}^2} = \frac{\sqrt{\mu_\Omega(\Omega)}}{|\lambda_j|}$. Thus we get $\max_{j=1,\dots,m} \|\psi_j\|_{G_\phi} \leq \frac{\sqrt{\mu_\Omega(\Omega)}}{|\lambda_m|}$. So by Theorem 4, $d_n(B_1(\|\cdot\|_{G_\phi})) = d_n(G_\phi) \geq \frac{|\lambda_m|}{\sqrt{\mu_\Omega(\Omega)}} \sqrt{1 - \frac{n}{m}}$. \square

Note that for $m = n + 1$, the lower bound $\frac{|\lambda_{n+1}|}{\sqrt{\mu_\Omega(\Omega)}} \sqrt{1 - \frac{n}{n+1}} = \frac{|\lambda_{n+1}|}{\sqrt{\mu_\Omega(\Omega)}} \sqrt{\frac{1}{n+1}}$ from Theorem 7(ii) is smaller than the lower bound $\frac{|\lambda_{n+1}|}{\sqrt{\mu_\Omega(\Omega)}}$ from Corollary 11.

Thus the argument using singular numbers gives better results than the one combining an inclusion of orthonormal subsets with estimates of variational norms based on Theorem 2. However, in cases when better estimates of G_ϕ -variations of eigenfunctions can be obtained, it can be verified that Theorem 7(i) provides larger lower bounds than the ones from Corollary 11.

Although it is not emphasized in the notation, the lower bounds on the n -width depend on the number of variables d . When $\mu = \mu_d$ is the d -dimensional Lebesgue measure, then the term $\mu_d(\Omega_d)$ in the lower bounds can either grow to infinity exponentially fast (e.g., when $\Omega_d = [-1, 1]^d$) or go to zero exponentially fast (e.g., when Ω_d is the unit Euclidean ball) or be constant (e.g., when $\Omega_d = [0, 1]^d$). So, for domains formed by Euclidean balls of high dimensions d , our estimates of the n -width can be large for those n for which $|\lambda_{n+1}|$ is large with respect to the square root of the volume of the d -dimensional unit ball. Note that kernels satisfying the assumptions of Corollary 11 are Hilbert-Schmidt, so $\sum_{j=1}^{\infty} \lambda_j^2 < \infty$. Hence $\lim_{n \rightarrow \infty} \frac{|\lambda_n|}{\sqrt{n}} = 0$, i.e., $|\lambda_n| = o(\frac{1}{\sqrt{n}})$ (see [22, p. 263]).

5 Comparisons of Worst-Case Errors

Combining the upper bounds on rates of approximation by $\text{span}_n G$ from Theorem 1 with the lower bounds on n -width from Section 4, we can compare worst-case errors in variable-basis and linear approximation. For any subset M of $\mathcal{L}_{\mu_\Omega}^2(\Omega)$ we denote by

$$\Delta_n(M) := d_n(M) - \delta(M, \text{span}_n G)$$

the difference between its n -width and its deviation from $\text{span}_n G$.

As $d_n(G) = d_n(B_1(\|\cdot\|_G))$, the same worst-case error (or nearly worst-case when infimum is not minimum) as the one in the ball $B_1(\|\cdot\|_G)$ must be achieved by some function in G . Moreover, $\delta(G, \text{span}_n G) = 0$ and thus $\Delta_n(G) = d_n(G) = d_n(B_1(\|\cdot\|_G))$. So we obtain from Theorems 1, 6, 7, and Corollary 1 the following estimates.

Corollary 2. *Let $\Omega \subseteq \mathbb{R}^d$, $Y \subseteq \mathbb{R}^q$, μ_Ω and μ_Y be σ -finite non degenerate Borel measures on Ω and Y , resp., $\phi \in \mathcal{L}_{\mu_\Omega \times \mu_Y}^2(\Omega \times Y)$, $G_\phi = \{\phi(\cdot, y) | y \in Y\}$ a bounded subset of $(\mathcal{L}_{\mu_\Omega}^2(\Omega), \|\cdot\|_{\mathcal{L}_{\mu_\Omega}^2})$ with $s_{G_\phi} = \sup_{y \in Y} \|\phi(\cdot, y)\|_{\mathcal{L}_{\mu_\Omega}^2}$, $\{\psi_j\}$ and $\{\lambda_j\}$ be eigenfunctions and corresponding eigenvalues of T_ϕ (ordered non increasingly in absolute values). Then for every positive integer n ,*

(i) for $\mu_Y(Y)$ finite

$$\Delta_n(G_\phi) \geq \frac{s_{n+1}(T_\phi)}{\sqrt{\mu_Y(Y)}}$$

and when $\Omega = Y$, $\mu_\Omega = \mu_Y$, and ϕ is symmetric

$$\Delta_n(G_\phi) \geq \frac{|\lambda_{n+1}|}{\sqrt{\mu_\Omega(\Omega)}} ;$$

(ii) for all m such that $m \geq n$ and for $c_{m, G_\phi} = \max_{j=1, \dots, m} \|\psi_j\|_{G_\phi}$,

$$\Delta_n(B_1(\|\cdot\|_{G_\phi})) \geq \frac{1}{c_{m, G_\phi}} \sqrt{1 - \frac{n}{m}} - \frac{s_{G_\phi}}{\sqrt{n}} ;$$

(iii) for $\mu_Y(Y)$ finite, every \mathcal{X}_n an n -dimensional linear subspace of $\mathcal{L}_{\mu_\Omega}^2(\Omega)$, every $\varepsilon > 0$, and every $f \in \text{conv } G_\phi$, there exists $\delta_f \in (0, 1]$ such that

$$\|f - \mathcal{X}_n\|_{\mathcal{L}_{\mu_\Omega}^2} - \|f - \text{conv}_n G_\phi\|_{\mathcal{L}_{\mu_\Omega}^2} \geq \frac{s_{n+1}(T_\phi)}{\sqrt{\mu_Y(Y)}} - \sqrt{(1 - \delta_f^2)^{n-1} (s_{G_\phi}^2 - \|f\|_{\mathcal{L}^2}^2)} - \varepsilon .$$

Corollary 2(i) shows that for every linear approximator and every computational unit defined by a function $\phi \in \mathcal{L}_{\mu_\Omega \times \mu_Y}^2(\Omega \times Y)$ such that G_ϕ is bounded, and every $\varepsilon > 0$, one can find a parameter y such that $\phi(\cdot, y)$ has in the linear approximation an error larger than $\frac{s_{n+1}(T_\phi)}{\sqrt{\mu_Y(Y)}} - \varepsilon$ or $\frac{|\lambda_{n+1}|}{\sqrt{\mu_\Omega(\Omega)}} - \varepsilon$. As mentioned above, the choice of the domain Ω leads to radically different estimates. When the domain is the d -dimensional ball, then the lower bound grows exponentially fast with d , while for the cube $[-1, 1]^d$ it converges exponentially fast to zero.

For kernels for which $s_{n+1}(T_\phi)$ converges to zero slower than $\alpha^{(n-1)/2}$ with some $\alpha \in (0, 1)$, Corollary 2(iii) implies that for every function in the convex hull of G_ϕ and for sufficiently large n , approximation from the dictionary G_ϕ guarantees smaller errors than any given linear approximator.

The lower bound $\frac{1}{c_{m, G_\phi}} \sqrt{1 - \frac{n}{m}} - \frac{s_{G_\phi}}{\sqrt{n}}$ in Corollary 2(ii) is positive for operators T_ϕ whose eigenfunctions have small G_ϕ -variations compared to s_{G_ϕ} , such as $\psi_j(x) = \text{sinc}(x - j)$ (recall that $\text{sinc}(x) = \frac{\sin \pi x}{\pi x}$ for $x \neq 0$ and $\text{sinc}(0) = 1$); see [23] for details.

In some cases, one can find other large orthogonal subsets in balls in G_ϕ -variation. For perceptron networks, such orthogonal sets were described in [5].

6 Discussion

We have investigated model complexity requirements in neural-network and linear approximation. We took advantage of recent results on approximation from a dictionary which imply characterization of sets with good properties in neural-network approximation. For these sets, we derived lower bounds on worst-case errors in linear approximation. We compared these bounds with upper bounds on worst-case errors in approximation from a dictionary. We obtained conditions under which these upper bounds are smaller than lower bounds for all linear approximators. The conditions are given in terms of properties of ϕ , number d of network inputs, and volumes of the d -dimensional domains where functions to be approximated are defined. Note that the condition that $\Delta_n(B_1(\|\cdot\|_{G_\phi}))$ is positive is rather strong as it implies that networks with units computing ϕ outperform *any* linear approximator. However, neural networks might be advantageous even in cases when theoretically a better linear approximator might exist, because finding such a linear approximator might be an unfeasible task.

Acknowledgements

G. G. and M. S. were partially supported by a PRIN grant from the Italian Ministry for University and Research, project “Adaptive State Estimation and Optimal Control”. V. K. was partially supported by MŠMT grant INTELLI 0C10047 and the Institutional Research Plan AV0Z10300504. Collaboration of V.K. with M. S. and G. G. was partially supported by CNR - AV ČR project 2010-2012 “Complexity of Neural-Network and Kernel Computational Models”.

References

1. Kainen, P.C., Kůrková, V., Vogt, A.: Best approximation by Heaviside perceptron networks. *Neural Networks* 13, 695–697 (2000)
2. Kainen, P.C., Kůrková, V., Vogt, A.: Geometry and topology of continuous best and near best approximations. *J. of Approximation Theory* 105, 252–262 (2000)
3. Kainen, P.C., Kůrková, V., Vogt, A.: Continuity of approximation by neural networks in L_p -spaces. *Annals of Operational Research* 101, 143–147 (2001)
4. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. on Information Theory* 39, 930–945 (1993)
5. Kůrková, V., Sanguineti, M.: Comparison of worst case errors in linear and neural network approximation. *IEEE Trans. on Information Theory* 48, 264–275 (2002)
6. Pisier, G.: Remarques sur un résultat non publié de B. Maurey. In: Séminaire d’Analyse Fonctionnelle 1980-1981, École Polytechnique, Centre de Mathématiques, Palaiseau, France, vol. I(12) (1981)

7. Jones, L.K.: A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Annals of Statistics* 24, 608–613 (1992)
8. Kůrková, V., Sanguineti, M.: Geometric upper bounds on rates of variable-basis approximation. *IEEE Trans. on Information Theory* 54, 5681–5688 (2008)
9. Gribonval, R., Vandergheynst, P.: On the exponential convergence of matching pursuits in quasi-incoherent dictionaries. *IEEE Trans. on Information Theory* 52, 255–261 (2006)
10. Kolmogorov, A.N.: Über die beste Annäherung von Funktionen einer gegebenen Funktionenklasse. *Annals of Mathematics* 37(1), 107–110 (1936)
11. Kůrková, V.: Dimension-independent rates of approximation by neural networks. In: Warwick, K., Kárný, M., eds.: Computer-Intensive Methods in Control and Signal Processing. The Curse of Dimensionality, pp. 261–270. Birkhäuser, Boston (1997)
12. Barron, A.R.: Neural net approximation. In: Proc. 7th Yale Workshop on Adaptive and Learning Systems, pp. 69–72. Yale University Press, New Haven (1992)
13. Kůrková, V.: High-dimensional approximation and optimization by neural networks. In: Suykens, J., Horváth, G., Basu, S., Micchelli, C., Vandewalle, J. (eds.) *Advances in Learning Theory: Methods, Models and Applications*, ch. 4, pp. 69–88. IOS Press, Amsterdam (2003)
14. Kůrková, V.: Model complexity of neural networks and integral transforms. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) *ICANN 2009. LNCS*, vol. 5768, pp. 708–718. Springer, Heidelberg (2009)
15. Girosi, F., Anzellotti, G.: Rates of convergence for radial basis functions and neural networks. In: Mammone, R.J. (ed.) *Artificial Neural Networks for Speech and Vision*, pp. 97–113. Chapman & Hall, Boca Raton (1993)
16. Kůrková, V., Kainen, P.C., Kreinovich, V.: Estimates of the number of hidden units and variation with respect to half-spaces. *Neural Networks* 10, 1061–1068 (1997)
17. Kainen, P.C., Kůrková, V.: An integral upper bound for neural network approximation. *Neural Computation* 21, 2970–2989 (2009)
18. Kainen, P.C., Kůrková, V., Vogt, A.: A Sobolev-type upper bound for rates of approximation by linear combinations of Heaviside plane waves. *J. of Approximation Theory* 147, 1–10 (2007)
19. Kainen, P.C., Kůrková, V., Sanguineti, M.: Complexity of Gaussian radial basis networks approximating smooth functions. *J. of Complexity* 25, 63–74 (2009)
20. Pinkus, A.: *n*-Widths in Approximation Theory. Springer, Heidelberg (1985)
21. Shubin, M.A.: Pseudodifferential operators and spectral theory. Springer, Heidelberg (2001)
22. Birman, M.S., Solomyak, M.Z.: Spectral Theory of Self-Adjoint Operators in Hilbert Space. D. Reidel Publishing, Dordrecht (1987)
23. Gnecco, G., Kůrková, V., Sanguineti, M.: Some comparisons of complexity of dictionary-based and linear computational models (2010) (manuscript)

A Funny Proverb Generation System Based on *Sukashi*

Hiroaki Yamane and Masafumi Hagiwara

The Department of Information and Computer Science, Keio University,
Hiyoshi 3-14-1, Kohoku-ku, Yokohama, 223-8522 Japan
{yamane,hagiwara}@soft.ics.keio.ac.jp

Abstract. In this paper, we propose a system which produces funny proverbs. This system uses the punch line framework named *Sukashi*. That is, by changing the end of the line, the proposed system produces a funny sentence. In this system, we employ Google N-grams to make a lot of *Sukashi* candidates. After that, the system extracts parameters from each word in each candidate. We choose parameters such as words' sounding, length, imageability, similarity and concrete level. The system selects candidates by using fuzzy rules. The performance of the proposed system has been evaluated by subjective experiments and obtained satisfactory results.

Keywords: Laugh, Text generation, Fuzzy rules, *Sukashi*.

1 Introduction

Laughing is an essential element for human beings. It makes people's relationships better owing to humor [1]. Moreover, human beings are the species that look for funny things. Putting humor which raises a good laugh is one of the most intelligent activities by human beings. Advancing this field not only contributes to deeper understanding of humankind, but also constructs more human-friendly interface. From various kinds of fields such as philosophy, literature, psychology and entertainment, etc., people has been studying laugh. For engineered approaches, "pun generation" has been a big topic [2][3]. However, as far as we know, few studies actually focused on the funny level of generated items. To deal with this problem, we focus on a kind of Japanese punch line named *Sukashi* [4].

We show the structure of *Sukashi* in Fig. 1. Owing to *Sukashi*, we are able to make these funny proverbs in simpler and more flexible ways. In this paper, we propose a system which produces funny proverbs by using *Sukashi*. The remainder of the paper is organized as follows. We first describe the *Sukashi* generation system in Section. 2. Then we show experimental results in Section. 3. Finally, we conclude with the discussion and directions for future work in Section. 4.

2 An Automatic Funny Proverb Generation System Based on *Sukashi*

We summarize the flow of the system in Fig. 2

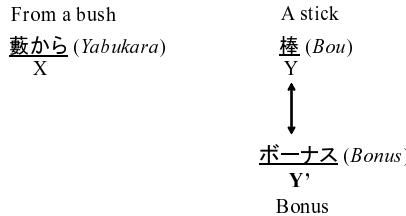


Fig. 1. Structure of *Sukashi*. The meaning of the original proverb is “out of the blue” and is directly translated as “a stick from a bush.” By changing the end word “Bou” to “Bonus”, it becomes funnier.

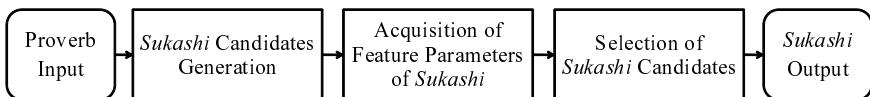


Fig. 2. Overall view of the proposed system

The proposed system consists of following five steps. That is, 1. Proverb Input 2. *Sukashi* Candidates Generation 3. Acquisition of Feature Parameters of the Candidates 4. Selection of *Sukashi* Candidates 5. *Sukashi* Output. We will explain each step in more detail.

2.1 *Sukashi* Candidates Generation

In order to generate *Sukashi* candidates, the input proverbs are divided into two parts, *X* and *Y*. We divide them by the difference of part of speech. For example, in Fig. 1, *X* is before noun “bou” and the rest is *Y*. The proposed system checks the input proverbs whether each part of them has a noun or a verb using Japanese morphological analyzer called MeCab [5], then checks whether the last of part of speech has *Kanji*, the system extracts *X* from them. After that, for these *X* the system finds sentences which are suitable for them to concatenate by using Japanese Google N-grams Corpus [6].

2.2 Acquisition of Feature Parameters of the Candidates

The following five parameters are employed in the proposed system.

- a) The number of characters and accordance of vowels in punch lines
- b) Difference of sounds
- c) Imageability
- d) Similarity of words
- e) Concrete level

In a), we employ the number of characters and accordance of vowels in punch line as a parameter. According to our preliminary study, people tend to evaluate

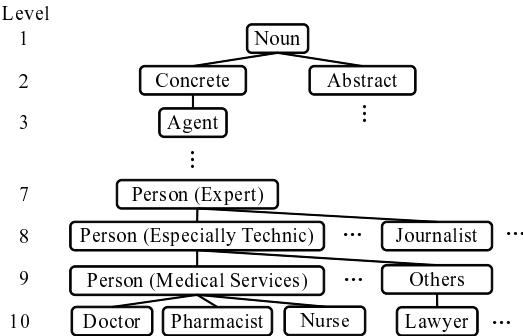


Fig. 3. Tree structure of Goi-Taikei. We see each word in each depth as concrete level.

Sukashi funnier if the head and the end vowels are the same as original ones. – If these *Sukashis* are romanized, the number of characters in Y' is less than that of Y in the range of 1. – We set up the rule that if *Sukashi* candidates match these conditions, the system assumes that they are funny.

In b), we use difference of sounds as a parameter. As Oda [7] points out, if things are funny they may have points of similarity. We calculated the difference of sounds by romanizing *Sukashi* and using Dynamic Programming [8].

In c), imageability is selected. Through the preliminary study, a number of subjects commented things are funny because they are easy to imagine. In addition, from brain science field, high-imageability nouns activate large part of human's brain [9]. We used NTT Data Series Imageability dictionary [10] and extracted a parameter.

In d), we employ a parameter, which deals with similarity of words. Oda points out that “sudden change of idea or behavior” and “unjustified expansion of idea or behavior” are essential elements for laughing. Therefore, a punch line needs to be different from original one. In the proposed system, by using Computational System for the Similarity between Concepts(Words) [11]-[13], values are calculated.

In e), concrete level is employed as a parameter. Owing to the preliminary study, we assumed if things become more concrete, they will be funnier. We display a tree structure of Japanese word-relation dictionary, Goi-Taikei [14] in Fig. 3. Hence, with measuring the depth of the tree structure in Goi-Taikei, concrete levels are granted.

2.3 Selection of *Sukashi* Candidates

Fig. 4 summarizes the flow of the selection of *Sukashi* candidates.

First, the system acquires each parameter from each word in the punch line of *Sukashi* candidates. In addition, if there are multiple values, the maximum one is selected. As Fig. 4 shows after acquiring values, fuzzy rules are applied. There are two kinds of fuzzy rules indicated by Fuzzy Rules A and Fuzzy Rules B. After

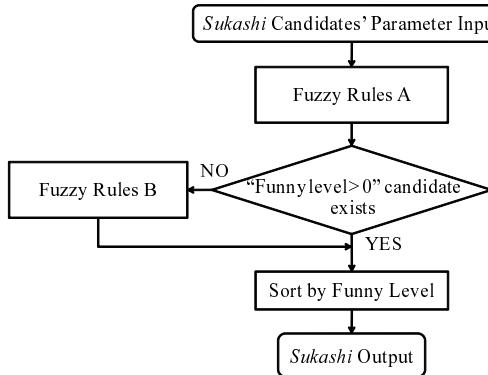


Fig. 4. Flow of fuzzy parts of the proposed system

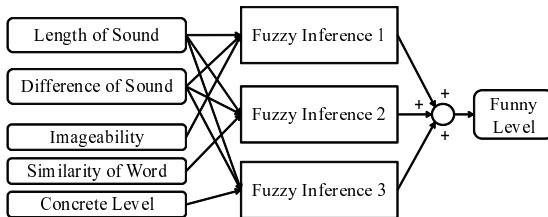


Fig. 5. Fuzzy Rule A of the proposed system

applying the first one, the system outputs sorted *Sukashi* candidates if there is one or more *Sukashis* whose Funny level is greater than zero. If not, Fuzzy Rules B is applied, which means the output is going to be the other sorted *Sukashi* candidates. We employ if-then rule [15] with direct method for calculation. Fuzzy Rules A has 2 antecedent parts and B has 3. With centroid computation for the grades, final output values are calculated to estimate funny level such as “not so funny” and “funny”.

Fig. 5 represents the flow of the Fuzzy Rules A.

Rules are

- Rule A1
If “Y is long” and “Difference of sounds is small” and “Imageability is high” then “*Sukashi* is funny”
- Rule A2
If “Y is long” and “Difference of sounds is small” and “Similarity of words is related term degree” then “*Sukashi* is funny”
- Rule A3
If “Y is long” and “Difference of sounds is small” and “Concrete level is high” then “*Sukashi* is funny”

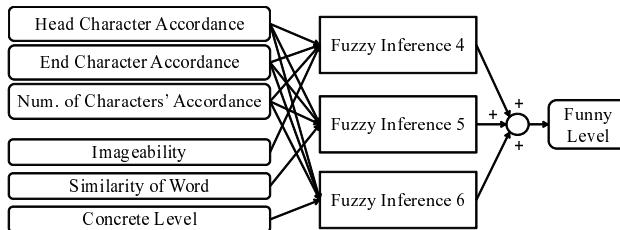


Fig. 6. Fuzzy Rule B of the proposed system

- Rule A4
If “Y is middle long” and “Difference of sounds is small” and “Imageability is high” then “*Sukashi* is somewhat funny”
- Rule A5
If “Y is middle long” and “Difference of sounds is small” and “Similarity of words is *related term degree*” then “*Sukashi* is somewhat funny”
- Rule A6
If “Y is middle long” and “Difference of sounds is small” and “Concrete level is high” then “*Sukashi* is somewhat funny”

We show the flow of the Fuzzy Rules B in Fig. 6. Here, the rules are given priority in order of Rule B1-B3, B4-B6, B7-B9.

- Rule B1
If “Head of Y and that of Y' are the same” and “Imageability is high” then “*Sukashi* is funny”
- Rule B2
If “Head of Y and that of Y' are the same” and “Similarity of words is *related term degree*” then “*Sukashi* is funny”
- Rule B3
If “Head of Y and that of Y' are the same” and “Concrete level is high” then “*Sukashi* is funny”
- Rule B4
If “End of Y and that of Y' are the same” and “Imageability is high” then “*Sukashi* is funny”
- Rule B5
If “End of Y and that of Y' are the same” and “Similarity of words is *related term degree*” then “*Sukashi* is funny”
- Rule B6
If “End of Y and that of Y' are the same” and “Concrete level is high” then “*Sukashi* is funny”
- Rule B7
If “Number of characters in Y and that of Y' are in the range of 1” and “Imageability is high” then “*Sukashi* is funny”

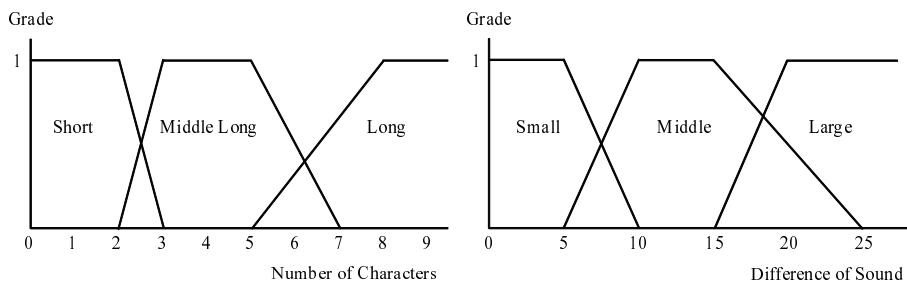


Fig. 7. Membership functions of Y's length and difference of sounds

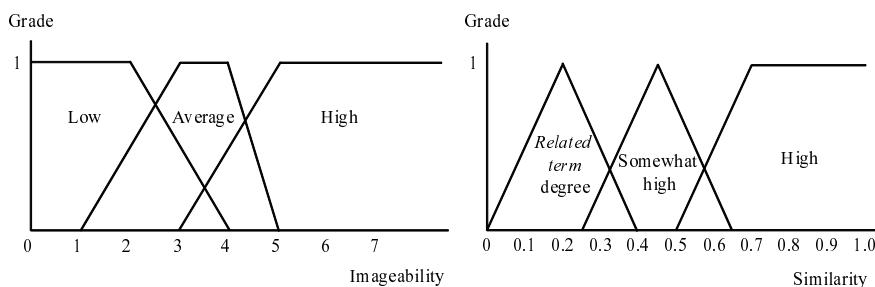


Fig. 8. Membership functions of imageability and similarity

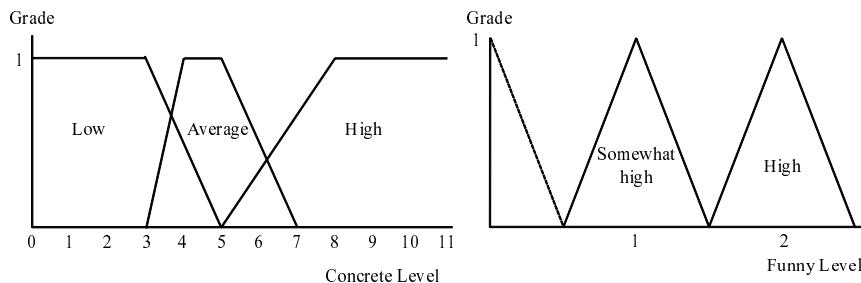


Fig. 9. Membership functions of concrete level and consequent part

– Rule B8

If “Number of characters in Y and that of Y' are in the range of 1” and “Similarity of words is *related term degree*” then “*Sukashi* is funny”

– Rule B9

If “Number of characters in Y and that of Y' are in the range of 1” and “Concrete level is high” then “*Sukashi* is funny”

We display the membership functions of length of Y and difference of sounds in Fig. 7, imageability and similarity in Fig. 8, concrete level and consequent part in Fig. 9.

3 Experiments

We conducted verification experiments to examine funny level of each generated *Sukashi* by our system. The experiments were performed in terms of funny level and unpredictable quality.

Funny Level

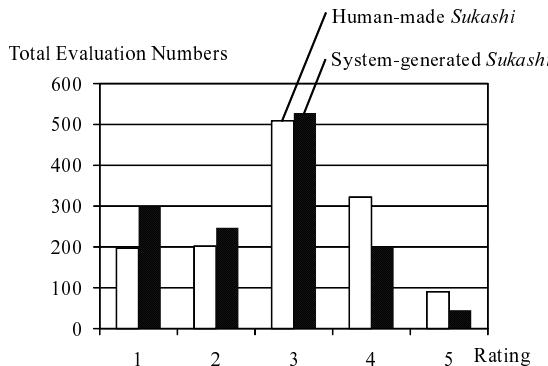
- 5 points: *Sukashi* is funny
- 4 points: *Sukashi* is somewhat funny
- 3 points: *Sukashi* is average
- 2 points: *Sukashi* is somewhat boring
- 1 point: *Sukashi* is boring or not understandable

Unpredictable Quality

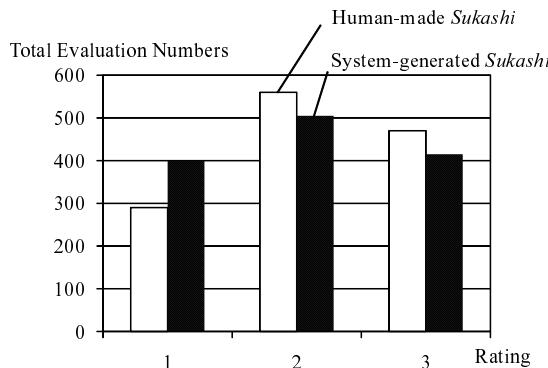
- 3 points: Punch line of *Sukashi* is beyond expectation
- 2 points: Punch line of *Sukashi* is predictable if you assume it is *Sukashi*
- 1 point: Punch line of *Sukashi* is predictable because it is a normal sentence

Table 1. Examples of system-generated *Sukashis*

Proverb	<i>Sukashi</i>
鬼に金棒: An ogre with an iron club <i>Oni ni Kanabou</i> (Unbeatable advantage)	→ 鬼にカネボウ: An ogre with “Kanebo” <i>Oni ni Kanebou</i> “Kanebo” is a famous cosmetic company’s product
漁夫の利: The fisherman’s profit <i>Gyofu no Ri</i> (Gaining the third party’s profit gain)	→ 漁夫のリハビリ: The fisherman’s rehabilitation <i>Gyofu no Rihabiri</i>
縁の下の力持ち: The strong person under the floor <i>Ennoshita no Chikaramochi</i> (The strong person working in the background)	→ 縁の下の父: The father working in the background <i>Ennoshita no Chichi</i>
年寄りの冷や水: The cold water of an old person <i>Toshiyori no Hiyanizu</i> (A hustling old man to keep up with the young)	→ 年寄りの鼻水: The snot of an old man <i>Toshiyori no Hanamizu</i>
焼け石に水: Sprinkling water to a burnt stone <i>Yakeishi ni Mizu</i> (Only a drop in the ocean)	→ 焼け石に傷: Do hurt to a burnt stone <i>Yakeishi ni Kizu</i>
身から出た錆: The rust which comes out of a body <i>Mi kara Deta Sabi</i> (An ill life, an ill end)	→ 身から出たわさび: Wasabi which comes out of a body <i>Mi kara Deta Wasabi</i> “Wasabi” is Japanese spice
塞翁が馬: The horse of an old man living in the fort <i>Saiou ga Uma</i> (Good can come out of a misfortune)	→ 塞翁がネタ: A old man living in the fort is newsworthy <i>Saiou ga Neta</i>
知らぬが仏: The Buddha who doesn’t know the truth <i>Shiranu ga Hotoke</i> (Ignorance is bliss)	→ 知らぬがほっとけよう: I don’t know but leave me alone <i>Shiranu ga Hotokeyou</i>

**Fig. 10.** Comparison of funny level**Table 2.** Difference of Funny level features

	Human-made <i>Sukashi</i>	System-generated <i>Sukashi</i>
Funny level on average	2.93	2.58
Variance of funny level	1.19	1.08

**Fig. 11.** Comparison of unpredictable quality

3.1 Experiment Condition

We used 50 proverbs [16] as input for the system. In order to evaluate the quality of the generated *Sukashis*, we carried out *Turing test* like experiments. The system generated 291 *Sukashis*. We shuffled both of system-generated ones and human-made ones randomly on condition that the number of *Sukashis* for each proverb is the same. Subjects evaluated 240 *Sukashis* in terms of funny level and unpredictable quality without knowing which is the system-generated one. The number of subjects is 11. Table 1 illustrates some examples of *Sukashis* which were produced by the proposed system.

Table 3. Difference of unpredictable quality features

	Human-made <i>Sukashi</i>	System-generated <i>Sukashi</i>
Unpredictable quality on average	2.14	2.01
Variance of unpredictable quality	0.563	0.613

3.2 Experimental Result

Fig. 10 is the result of comparison with the funny level of system-generated *Sukashis* and human-made ones. As the figure demonstrates the system-generated *Sukashis* are comparable to the human-made ones. Table 2 shows funny features of both of them.

Fig. 11 represents the result of comparison with the unpredictable quality of system-generated *Sukashis* and human-made ones. Unpredictable features of them is shown in Table 3.

4 Conclusion

In this paper, we proposed a new system which produces funny proverbs. The proposed system uses the punch line framework named *Sukashi*. That is, by changing the end of the line, it produces a funny sentence. In the proposed system, we have employed Japanese Google N-grams to make a lot of *Sukashi* candidates. After that, the system extracts parameters from each word in each candidate. We choose parameters such as words' sounding, length, imageability, similarity and concrete level. Then the system selects candidates by using fuzzy rules with centroid computations.

The performance has been evaluated by subjective experiments and they shows that the system-generated *Sukashis* are comparable to human-made ones.

Finding a laughing framework itself – named *Sukashi* – is the first step. We believe the direction of this research will contribute to construct more human-friendly interfaces such as communication and entertainer robots.

Acknowledgments. We express our deepest appreciation for Prof. Tsutomu Ishikawa's provision of the Computational System for the Similarity between Concepts(Words). Also we are grateful to Google and Gengo-Shigen-Kyokai (GSK) for Japanese Google N-grams.

References

1. De Boni, M., Richardson, A., Hurling, R.: Humour, Relationship Maintenance and Personality Matching in Automated Dialogue: A Controlled Study. *Interact. Comput.* 20(2), 342–353 (2007)
2. Binsted, K., Takizawa, O.: BOKE – A Japanese Punning Riddle Generator. *Journal of the Japanese Society for Artificial Intelligence* 13(6), 920–927 (1997)

3. Waller, A., Black, R., O'Mara, D.A., Pain, H., Ritchie, G., Manurung, R.: Evaluating the STANDUP Pun Generating Software with Children with Cerebral Palsy. *ACM Trans. Access. Comput.* 1(3), 1–27 (2009)
4. Fukui, N.: The Techniques for Making Laughter – Laugh Makes Us Discover the World, Sekai Shisousya (2002)
5. Yet Another Part-of-Speech and Morphological Analyzer,
<http://mecab.sourceforge.net/>
6. Kudo, T., Kazawa, H.: Japanese Google N-grams, vol.1. Gengo-Shigen-Kyokai (GSK)
7. Oda, S.: Laughing and Humor. Chikuma Shobo (1986)
8. Needleman, S.B., Wunsch, C.D.: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology* 48(3), 443–453 (1970)
9. Sabsevitz, D., Medler, D., Seidenberg, M., Binder, J.: Modulation of the Semantic System by Word Imageability. *Neuro Image* 27(1), 188–200 (2005)
10. Sakuma, N., Ijuin, M., Fushimi, T., Tatsumi, I., Tanaka, M., Amano, S., Kondo, K.: “NTT Database Series” Japanese Vocabulary Attribution. *Word Imageability* 8(1) (2005)
11. Ishikawa Laboratory, Takushoku University, Computational System for the Similarity between Concepts(Words),
http://www.cs.takushoku-u.ac.jp/ai/ruiji/Similarity_System.cgi
12. Noguchi, Y., Shimizu, R., Sugimoto, K., Ishikawa, T.: An Improved Tool for Measuring Semantic Similarity between Words. In: The 69th National Convention of Information Processing Society of Japan, vol. 2, pp. 2545–2546 (March 2007)
13. Kawashima, T., Ishikawa, T.: An Evaluation of Knowledge Base of Words and Thesaurus on Measuring the Semantic Similarity between Words. In: The 18th Annual Conference of the Japanese Society for Artificial Intelligence, vol. 18, pp. 2D2–2D10 (2004)
14. Shoten, I.: Nihongo Goi-Taikei CD-ROM,
<http://www.kecl.ntt.co.jp/mtg/resources/GoiTaikei/>
15. Sugeno, M.: Fuzzy Control. Nikkan Kogyo (1988)
16. Kurogo, T.: Kurogo's Proverb Dictionary, <http://www.geocities.jp/tomomi965/>

Supervised Neural Fuzzy Schemes in Video Transmission over Bluetooth^{*,**}

Hassan B. Kazemian, Guillaume F. Remy, and Phil Picton

London Metropolitan University
Faculty of Computing, London, United Kingdom
h.kazemian@londonmet.ac.uk

Abstract. This paper is concerned with an intelligent application of Moving Picture Expert Group (MPEG) video transmission in a Bluetooth link. MPEG Variable Bit Rate (VBR) video is data hungry and results in excessive data loss and time delay over wireless networks. In a Bluetooth channel transmission rate is unpredictable as result of interferences by channel noises or other close by wireless devices. Therefore, it is difficult to transmit MPEG VBR video in Bluetooth without data loss and/or time delay. A buffer entitled ‘traffic regulating buffer’ is introduced before the Bluetooth protocol stack to prevent excessive overflow of MPEG video data over the network. Two novel neural fuzzy schemes are presented in this paper to monitor the traffic regulating buffer input and output rates, in order to ensure the video stream conforms to the traffic conditions of Bluetooth. The computer simulation results demonstrate that the two supervised neural fuzzy schemes reduce standard deviation and data loss, when compared with a traditional MPEG VBR video transmission over Bluetooth.

Keywords: Hybrid supervised neural fuzzy scheme, MPEG video, Bluetooth.

1 Introduction

MPEG video compression utilizes the similarities and redundancies between consecutive frames in order to reduce the size of video stream. In Variable Bit Rate (VBR) coding, complex and complicated scenes usually require more data and hence demand a larger bandwidth for transmission [1]. This kind of video coding offers constant video quality by statically setting the quantization scale for I- (Intra), P- (Predicted) and B- (Bi-directional) frames while the bit rate is allowed to vary. This paper uses neural fuzzy scheme to guarantee that the VBR coding from the host satisfies the traffic condition of the Bluetooth wireless channel during the transmission period.

Fuzzy logic deals with imprecision, ambiguity and vagueness of the real world problems using simple linguistic statements and by this means achieves better performance, robustness and low solution cost [2]. Therefore, fuzzy logic has been

* This research was funded by EPSRC, UK.

** The work was carried out at London Metropolitan University, UK.

applied to video transmission over wireless technology. For instance, fuzzy logic has been applied to wireless LAN. By using fuzzy logic and internal model control, the system adapts the bit rate, reduces power consumption and satisfies the Quality of Service (QoS) requirements [3]. In another example, A fuzzy logic-based method for assessing and improving audio and video QoS in IEEE 802.11 wireless networks has been developed. By using fuzzy-based method, the average QoS for both audio and video applications was significantly improved compared with the conventional IEEE 802.11 protocol [4]. There have also been some applications of neural networks to signal processing [5] and video transmission in wireless technology. For example, a neural network system is developed to predict MPEG-coded video source traffic for more efficient dynamic bandwidth management and improved QoS. Experiential results demonstrate that recurrent and feed-forward neural networks are comparable and sometimes even better than the results reported in the literature [6]. In another study a neural network has been applied to bandwidth prediction and control for transmission of real-time video stream over mobile ad hoc network (MANET). The experimental results reveal that there are considerable improvements in the packet loss and real time visual quality as compared with conventional video transmission over MANET [7]. Furthermore, an artificial neural network has been applied to voice, video and web traffic management and scheduling in wireless ATM and WiFi networks to improve QoS. The results obtained in scheduling the traffic shows a significant capacity improvement using the neural networks as compared with other techniques proposed in the literature [8].

The research in the applications of fuzzy logic and neural networks have been taken further and hybrid neural fuzzy have been used to improve video quality over wireless networks. In a research work, a fuzzy neuron network scheme capable of adjusting input and output is introduced to increase robustness, stability and working speed of the network for general packet radio service congestion control. As shown by the experimental results, the algorithm significantly prevents the congestion for video transmission, which proves that the method is more practical and effective than traditional methods for congestion control in wireless networks [9]. In another research a neural fuzzy controller is proposed to automatically determine the quantization scale for each Group Of Picture (GOP) of the encoder during the transmission period. Computer simulation results show that the proposed improved quantization scale increases transmission rate and reduces data loss, excessive delay and image quality degradation as compared with a conventional quantization scale in 2.4 GHz frequency band [10]. This paper takes the research further by applying two supervised neural fuzzy controllers to video transmission over Bluetooth. This paper deals with the overall performance of the two supervised neural fuzzy schemes in the Bluetooth channel with presence of noise.

2 MPEG Video

Fig. 1 represents the block diagram of a MPEG encoder utilized in this research. A MPEG video sequence consists of a number of GOPs, which is a repeated pattern of I, P and B frames. The P and B frames first pass through a temporal redundancy compression. In Fig. 1, temporal redundancy compression consists of motion compensation and motion estimation [11]. In the temporal redundancy model, the reference

frame is used to attain the motion vector. This is referred to as motion estimation, where each block of B or P frame is explored against blocks within the reference frame to get the motion vector for each of them. Motion compensation of temporal redundancy compression presents a picture in terms of the transformation of a reference picture to the current picture. The reference picture can be previous in time or even from the future.

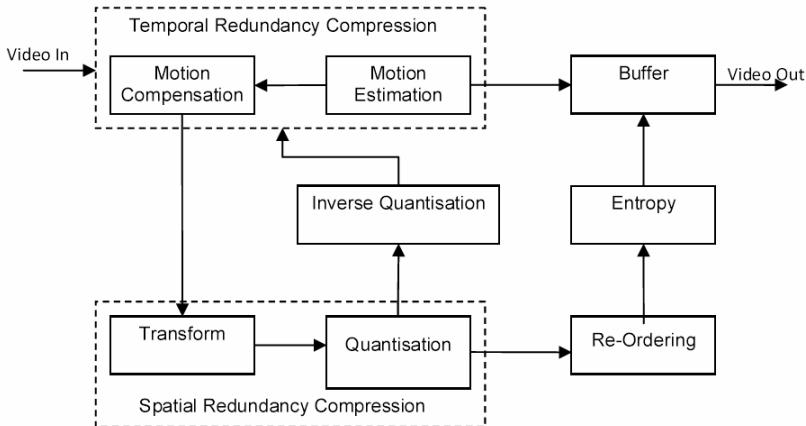


Fig. 1. MPEG encoder

Spatial redundancy compression comprises of transformation and quantization [12]. Discrete Cosine Transform is widely used as the first stage of compression of digital video pictures. A video signal is composed of many component frequencies, made up of cosine waves. By removing some of the higher frequency components one can send less information at the expense of some deterioration in image quality. Even after the transformation, we still have the same volume of data and most of the image is still carried through. The quantization will remove the redundant values. In this process probabilities of the symbols, which represent elements of video sequence are assigned. These probabilities are then used in order to produce a compressed bit sequence, which will be ready for transmission. In this work, ‘Huffman Coding’ scheme is implemented. To ensure good entropy coding of those quantized coefficients, a reorder should be perform. The Zigzag scanning is the most common way of reordering the coefficients of a block [13].

3 Supervised Neural Fuzzy System

In this research two novel hybrid neural fuzzy controllers and a traffic regulating buffer are inserted between the video encoder and the IEEE 802.15.1 Bluetooth channel. The traffic regulating buffer is a temporary storage which avoids undue overflow of MPEG video data in the Bluetooth channel. The traffic regulating buffer is an additional buffer to the generic cell rate algorithm offered by IEEE 802.15.1 Bluetooth.

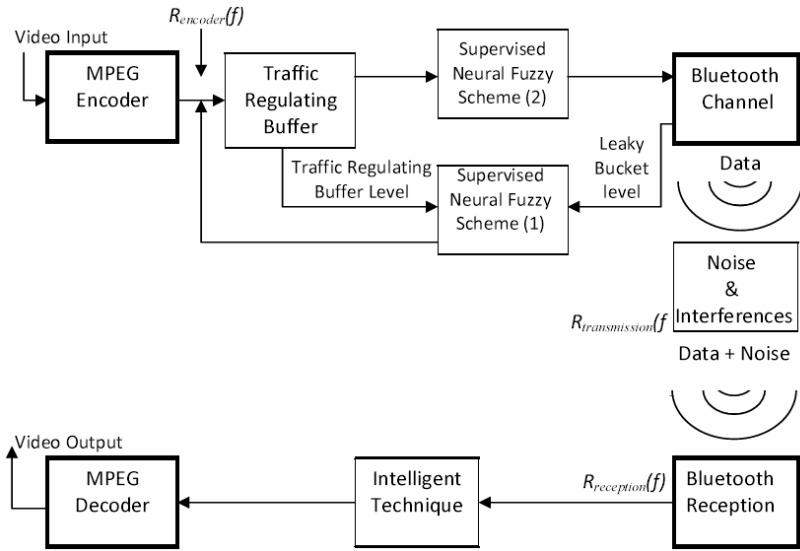


Fig. 2. The overall diagram of two neural fuzzy system

The generic cell rate algorithm in Bluetooth commonly known as leaky bucket, measures cell rate at a specified timescale [14]. However, traditional leaky bucket is mostly insufficient to control transmission of MPEG video data in Bluetooth [10].

Fig. 2 represents the overall diagram of supervised neural fuzzy system. MPEG encoder, traffic regulating buffer, supervised neural fuzzy (1) scheme, supervised neural fuzzy (2) scheme and Bluetooth channel are all at the sending port. Bluetooth reception, intelligent technique and MPEG decoder are at the receiving end. This paper only discusses the video transmission at the sending channel. Noise and interferences in Fig. 2 are represented by Gaussian noise. A distributed Gaussian noise is applied to the contracted bandwidth to simulate the actual video data rate at the point of transmission.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-np)^2}{2\sigma^2}} \quad (1)$$

where np is the mean value, p is the probability of an integer value of x , n is the number of events and σ is the standard deviation.

In Fig. 2, neural fuzzy (1) scheme supervises the traffic regulating buffer input rate to ensure the buffer is neither starved nor saturated of MPEG video data. Neural fuzzy (2) scheme is developed to smooth the video output in order to conform to the traffic conditions of the Bluetooth channel. In Fig. 3, the diagram of the hybrid neural fuzzy controller is presented. A Sugeno-type fuzzy controller [15] is utilized to create a direct mapping from the inputs and to the output of the neural fuzzy scheme. The first layer in Fig. 3 is the fuzzification layer. The activation function of a neuron is set to the membership function that specifies the neuron's fuzzy set. In this work, each neuron in fuzzification layer has a bell activation function, as described below [16].

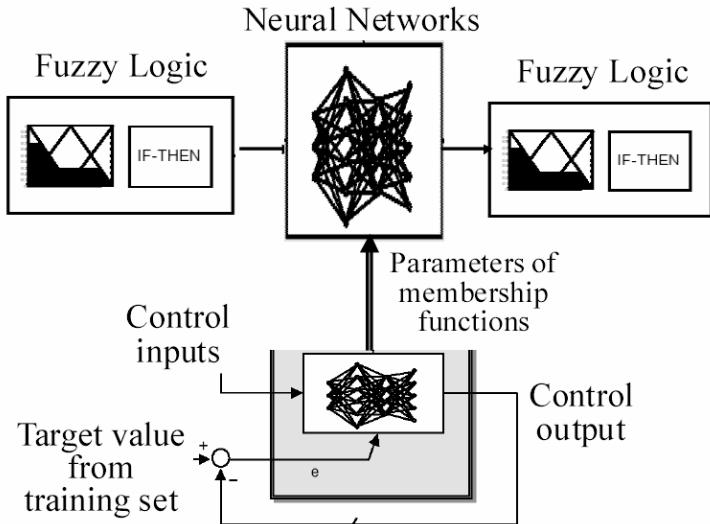


Fig. 3. Structure of supervised neural fuzzy

$$f(x) = \frac{1}{1 + \left| \frac{x - \varphi}{\delta} \right|^{2\beta}} \quad (2)$$

The shape of the bell varies by changing δ , β and φ parameters. The parameters δ and β both construct the shape of the membership function and the parameter φ determines the center of the curve; the parameter φ is usually positive. Each neuron in the second layer corresponds to a single Sugeno-type fuzzy rule. The third layer represents the defuzzification layer, which calculates the consequent value of each rule, weighted by the firing strength of the given rule. A zero-ordered Sugeno-type system is utilized for the neural fuzzy scheme, which means the activation function of each neuron in the defuzzification layer is equal to a constant. The values of these constants are determined through the training process of the neural fuzzy system. The output layer is obtained by a single summation neuron. This neuron calculates the sum of outputs of all defuzzification neurons and produces the overall output. An adaptive neural fuzzy inference system is trained to approximate the mapping created by neural fuzzy controller. Therefore, the adaptive neural fuzzy inference system has the same inputs and output as the hybrid neural fuzzy controller. Since the adaptive neural fuzzy inference system is fundamentally a multi-layered neural network, the neural fuzzy scheme is trained using the back-propagation algorithm [16].

4 Computer Simulation Results

The hybrid neural fuzzy system has been applied to a video clip entitled ‘Chicago’ [17]. Fig. 4 demonstrates the computer simulation results from the GOP 100 to the

GOP 180 with a bandwidth mean value of 724 Kbps. The total number of frames in this plot is 960. In Fig. 4, there are four different graphs, $R_{encoder}$, $R_{transmission}$, bandwidth with Gaussian noise, and superposition of $R_{encoder}$, $R_{transmission}$ and bandwidth for comparison purposes. $R_{encoder}$ is inputted to the proposed system and $R_{transmission}$ is the actual video transmission rate over the Bluetooth channel. $R_{transmission}$ show that the applications of the two hybrid neural fuzzy controllers to the traffic regulator buffer provide much smoother video output. Fig. 4 demonstrates that in terms of transmission rate, the supervised neural fuzzy system can manipulate the MPEG encoder as well as regulate the bit rate in order to keep the output in line with the Bluetooth channel. Therefore the burstiness of video data in $R_{transmission}$ is much more controlled for the Bluetooth channel resulting in reduced data loss and time delay, and better image quality. Moreover, Table 1 compares the standard deviation of $R_{encoder}$ the conventional open loop VBR system with the standard deviation of $R_{transmission}$ the proposed neural fuzzy system. The standard deviation for the hybrid neural fuzzy system is 88 Kbits/s much smaller than the conventional VBR system 263 Kbits/s, resulting in smoother video output. The figures demonstrate that the standard deviation is reduced by 66.54%, consequently more MPEG video data transmission, less data loss and improved picture quality.

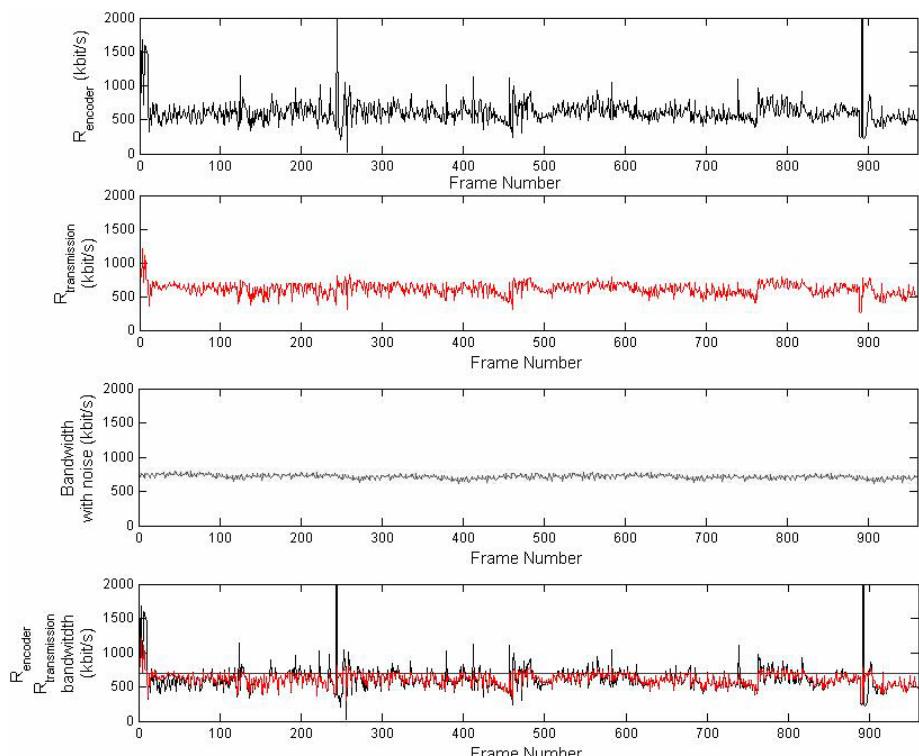


Fig. 4. Hybrid neural fuzzy system: data rate input $R_{encoder}$, data rate output $R_{transmission}$, bandwidth with noise and transposition for 960 frames

Table 1. Standard deviation comparison between hybrid neural fuzzy system and conventional open loop VBR system

	Hybrid neural fuzzy system	Conventional open loop VBR system
Standard deviation	88 kbps	263 kbps

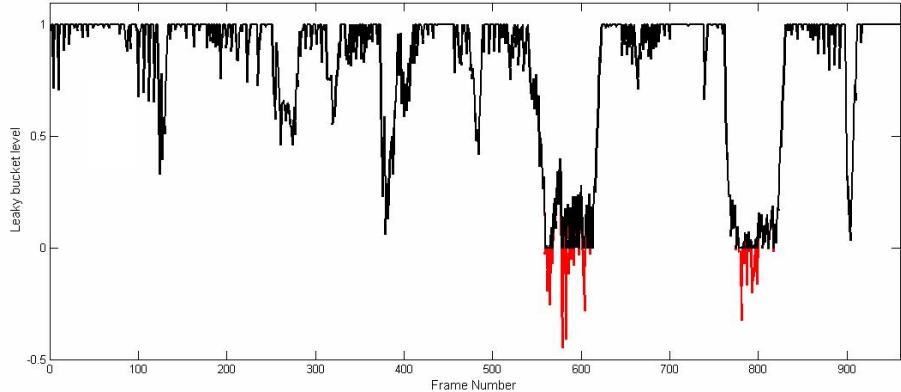


Fig. 5. Leaky bucket storage availability for conventional open loop VBR system for 960 frames

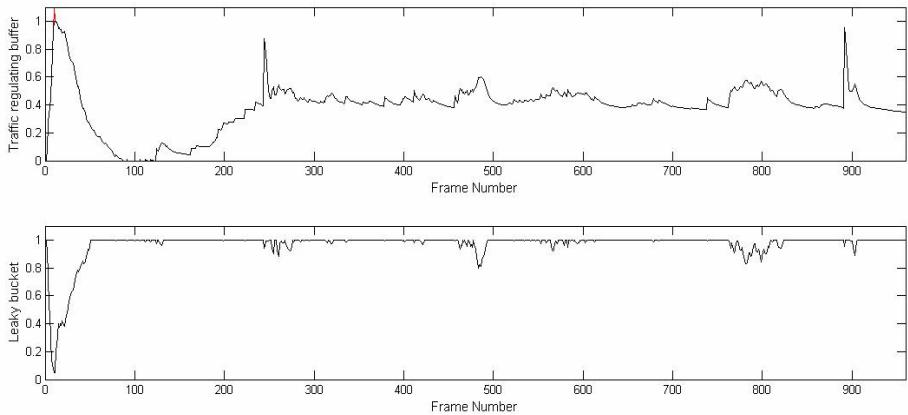


Fig. 6. Traffic regulating buffer and leaky bucket storage availabilities for the proposed neural fuzzy system for 960 frames

Figs. 5 and 6 further consider the capacity of the leaky bucket level and the traffic regulating buffer level. The leaky bucket algorithm works as such that the more token is taken the less storage capacity will be available. Therefore, level 1 means no token is used and 0 indicates all the tokens are used and the remaining data would be flushed out and lost. The traffic regulating buffer works in the other way round to the

leaky bucket, 0 shows no data is stored in the memory and 1 indicates that the buffer is full. Fig. 5 presents the leaky bucket for the conventional VBR MPEG video transmission over Bluetooth for 960 frames. The waveform section of the graph under zero shows the data loss and subsequently image quality degradation. In Fig. 5, no neural fuzzy techniques are implemented and therefore the some video data is inevitably flushed out and lost. Fig. 6 demonstrates the storage capacity for the traffic regulating buffer and the leaky bucket in the proposed neural fuzzy system. For this video clip, the introduction of the hybrid neural fuzzy control eliminates nearly all the data loss, as there is most of the time some storage capacity left in the traffic regulation buffer and the leaky bucket. Furthermore, by maintaining a sufficient space available in the leaky bucket, the novel neural fuzzy scheme decreases the standard deviation of the output bit rate and the number of dropped data, resulting in better picture quality and video stability.

5 Conclusion

This research work aims to design an artificial intelligence technique to improve video transmission over IEEE 802.15.1 Bluetooth. In this regard two novel supervised neural fuzzy controllers are developed to monitor the input and output of a traffic regulating buffer to guarantee that the MPEG video data conforms to the Bluetooth channel bandwidth. The two hybrid neural fuzzy decrease the standard deviation of the MPEG video and hence enable more data to be transmitted through the Bluetooth channel. Transmission of smoother video stream reduces data loss and time delay, which effectively generates video data stability and image quality at the receiving end. The neural fuzzy schemes also enhance the quality of picture with presence of noise over the Bluetooth channel. Finally the proposed two supervised neural fuzzy controllers smooth the traffic congestion and sustain constant video quality.

References

- Richardson, I.E.G.: H.264 and MPEG-4 Video Compression Video Coding for Next-Generation Multimedia. John Wiley and Sons Ltd., Chichester (2003), ISBN: 0470848375
- Zadeh, L.A.: Fuzzy Logic = Computing with Words. *IEEE Transaction on Fuzzy Systems* 4(2), 103–111 (1996)
- Pietrabissa, A., Razzano, G., Andreani, L.: Wireless LANs: an Advanced Control System for Efficient Power Saving. *Control Engineering Practice* 11(10), 1195–1207 (2003)
- Saraireh, M., Saatchi, R., Al-khayatt, S., Strachan, R.: Development and Evaluation of a Fuzzy Inference Engine System to Incorporate Quality of Service in IEEE 802.11 Medium Access Protocol. In: International Conference on Wireless and Mobile Communications, ICWMC 2006, Budapest, pp. 29–29 (2006)
- Mandic, D., Chambers, J., Haykin, S.(ed.): Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability. Wiley Series in Adaptive and Learning Systems for Signal Processing, Communications and Control Series (2001)
- Bhattacharya, A., Parlos, A.G., Atiya, A.F.: Prediction of MPEG-coded Video Source Traffic using Recurrent Neural Networks. *IEEE Transactions on Signal Processing* 51(8), 2177–2190 (2003)

7. Jiang, R., Tian, X., Chen, Y.: A New Bandwidth Prediction Based Wireless Ad-Hoc Video Communication. In: IET Conference Wireless, Mobile and Sensor Networks (CCWMSN 2007), Shanghai, China, pp. 1033–1038 (2007)
8. Fiengo, P., Giambene, G., Trentin, E.: Neural-Based Downlink Scheduling Algorithm for Broadband Wireless Networks. Computer Communications Journal 30, 207–218 (2007)
9. Su, J.L., Chen, Y., Ouyang, Z.: GPRS Congestion Control Algorithm Based on Fuzzy Kernel Neural Networks. In: ISDA 2006: Sixth International Conference on Intelligent Systems Design and Application, vol. 1, pp. 954–959 (2006)
10. Kazemian, H.B., Chantaraskul, S.: An Intelligent MPEG Rate Control in 2.4 GHz Frequency. International Journal of Adaptive Control and Signal Processing 24, 233–248 (2010)
11. Bhaskaran, V., Konstantinides, K.: Image and Video Compression Standards (1997), ISBN:0792399528
12. MPEG Tutorial-Introduction and Contents,
<http://www.bret1.com/mpeghtml/MPEGindex.htm>
13. Ghanbari, M.: Standard Codecs: Image Compression to Advanced Video Coding. In: IET, 2nd edn. (2010), ISBN-10: 086341964X, ISBN-13: 978-0863419645
14. Holt, A.: Network Performance Analysis: Using the J Programming Language. Springer, Heidelberg (2007), ISBN 1846288223, 9781846288227
15. User's guide, Fuzzy logic toolbox, for use with Matlab. The Math. Works Inc. (2008)
16. Negnevitsky, M.: Artificial Intelligence – a Guide to Intelligent Systems, 2nd edn. Addison-Wesley, Reading (2004)
17. Movie-List, <http://www.movie-list.com/nowplaying.shtml>

A Graph Based Framework for Clustering and Characterization of SOM

Rakia Jaziri¹, Khalid Benabdeslem², and Haytham Elghazel²

¹ University of Paris13 - LIPN CNRS 7030

99 Av J.B. Clement, 93430 Villeurbanne, France

² University of Lyon1 - LIESP EA 4125,

43 Bd du 11 Novembre, 69622 Villeurbanne, France

rakia.jaziri@lipn.univ-paris13.fr, {kbenabde,elghazel}@univ-lyon1.fr

Abstract. In this paper, a new graph based framework for clustering characterization is proposed. In this context, Self Organizing Map (SOM) is one popular method for clustering and visualizing high dimensional data, which is generally succeeded by another clustering methods (partitionnal or hierarchical) for optimizing the final partition. Recently, we have developed a new SOM clustering method based on graph coloring called McSOM. In the current study, we propose to automatically characterize the classes obtained by this method. To this end, we propose a new approach combining a statistical test with a maximum spanning tree for local features selection in each class. Experiments will be given over several databases for validating our approach.

Keywords: SOM, Characterization, Graphs.

1 Introduction

Clustering is an important task in knowledge discovery systems. It aims to discover the best structure from unlabelled data. To this end, the principle conducts a process of organizing objects into homogeneous groups. The clustering methods can be grouped into five families including partitioning, hierarchical, density-based, grid-based or model-based [1]. In this paper, we are interested in models and particularly those using self-organizing map (SOM) [13]. This technique is a prominent tool for high-dimensional data analysis since it provides a substantial data reduction that can be used for visualizing and exploring properties of data. Generally, two important issues need to be clarified after clustering data by SOM: how to optimize the number of neurons and how to characterize the obtained classes. To deal with the first issue, several authors have investigated SOM clustering in sequential ways [17] or simultaneous ones [3]. However these approaches don't take into account the topological neighborhood relations offered by SOM. For that reason, we present a recent developed method based on coloring of graphs where more details can be found in [6]. This method is called McSOM for Minimal coloring of SOM. It represents an extension of the minimal graph coloring technique for clustering.

After clustering task, it is important to select the optimal feature subset which is relevant for characterizing each obtained class. This task is generally done by experts or by automatic characterizing methods. In this context, several important research topics in cluster analysis and feature weighing are discussed in [2] [7] [8] [9]. For such approaches, the search for an optimal subset of features is built into the clustering construction making these techniques specific of a given learning algorithm. In [5] the authors discover the relevant features using filter techniques by looking only at the intrinsic property of data. The disadvantage is that this method ignores the interaction with the clustering algorithm and that most proposed techniques are univariate thus ignoring feature dependencies.

In contrast of these weighting algorithms, we propose a graph based approach for unsupervised feature selection. For each obtained class, our approach attempts to find a "Pivot" features using a statistical test and maps them into a maximum spanning trees for selecting their related subset of features that characterize this class. The rest of the paper is organized as follow: in section 2 we describe the batch version of SOM model. In section 3, we briefly present McSOM, a recent developed approach for clustering of SOM. The section 4 will be devoted to the used statistical test and maximum spanning trees for feature selection. Finally, we provide experimental results on several databases and a conclusion on our proposed work.

2 Self-Organizing Map: SOM

SOM is used nowadays through numerous domains and has been successfully applied in numerous applications. It is a very popular tool used for visualizing high dimensional data spaces. SOM can be considered as doing vector quantization and/or clustering while preserving the spatial ordering of the input data rejected by implementing an ordering of the codebook vectors (also called prototype vectors, cluster centroids or reference vectors) in a one or two dimensional output space. The SOM consists of neurons organized on a regular low-dimensional grid, called the map. More formally, the map is described by a graph (V, E) . V is a set of m interconnected neurons having a discret topology defined by E . For each pair of neurons (c, r) on the map, the distance $\delta(c, r)$ is defined as the shortest path between c and r on the graph. This distance imposes a neighborhood relation between neurons (Fig. II). Each neuron c is represented by a p -dimensional reference vector $w_c = \{w_c^1, \dots, w_c^p\}$ from \mathcal{W} (the set of all map's neurons), where p is equal to the dimension of the input vectors. The number of neurons may vary from a few dozen to several thousand depending on the application.

The SOM training algorithm resembles *k-means* [15]. The important distinction is that in addition to the best matching reference vector, its neighbors on the map are updated. The end result is that neighbouring neurons on the grid correspond to neighbouring regions in the input space.

The SOM algorithm is proposed on two versions: Stochastic (on-line) or batch (off-line) versions. In this paper, we use the second one which is deterministic and fast. The batch version of SOM is an iterative algorithm in which the whole

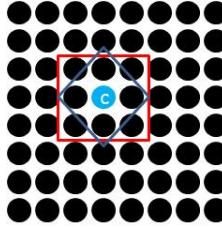


Fig. 1. Two dimensional topological map with 1-neighborhood of a neuron c . Rectangular (red) with 8 neighbors and diamond (blue) with 4 neighbors.

data set is presented to the map before any adjustments are made. In each training step, the data set is partitioned according to the Voronoi regions of the map reference vectors. More formally, we define an affectation function f from R^p (the input space) to C , that associates each element z_i of R^p to the neuron whose reference vector is “closest” to z_i (for the Euclidean distance). This function induces a partition $P = \{P_c; c = 1 \dots m\}$ of the set of individuals where each part P_c is defined by: $P_c = \{z_i \in \Omega; f(z_i) = c\}$. This represents the **assignment step**.

Next, an **adaptation step** is performed when the algorithm updates the reference vectors by minimizing a cost function, noted $\mathcal{E}(f, \mathcal{W})$. This function has to take into account the inertia of the partition P , while insuring the topology preserving property. To achieve these two goals, it is necessary to generalize the inertia function of P by introducing the neighborhood notion attached to the map. In the case of individuals belonging to R^p , this minimization can be done in a straight way. Indeed new reference vectors are calculated as:

$$w_r^{t+1} = \frac{\sum_{i=1}^n h_{rc}(t) z_i}{\sum_{i=1}^n h_{rc}(t)} \quad (1)$$

where $c = \arg \min_r \|z_i - w_r\|$, is the index of the best matching unit of the data sample z_i , $\|\cdot\|$ is the distance measure, typically the Euclidean distance, and t denotes the time. $h_{rc}(t)$ is the neighborhood kernel around the winner unit c . In practice, we often use $h_{rc} = e^{-\frac{\delta_{rc}}{2T^2}}$ where T represents the neighborhood radius in the map. It is decreased from an initial value T_{max} to a final value T_{min} .

Consequently, h_{rc} is a non-increasing function of time and of the distance of unit r from the winner unit c . The new reference vector is a weighted average of the data samples, where the weight of each data sample is the neighborhood function value $h_{rc}(t)$ at its winner c .

3 Graph Coloring of SOM: McSOM

In this section, we briefly present our recently developed graph based approach for clustering SOM. Let be $\Omega = \{z_1, z_2, \dots, z_n\}$ ($z_i \in R^p$) a data set clustered

with SOM into m neurons of a topological map. If we consider this map as an undirected graph $G(V, E)$ where V is a set of vertices (neurons) and E is a set of edges (similarities between neurons), we can use the graph theory methods for clustering SOM neurons. The main idea is to use neighbourhood relations to constrain the construction of the *threshold graph* (indispensable for the coloring algorithm) and the possible vertex selections during the building of the minimal coloring of this graph.

The *minimal coloring based clustering approach* requires a *non complete edge-weighted graph* $G_{>\theta} = (V, E_{>\theta})$ to return a partition P_k of $\mathcal{W} = \{w_1, w_2, \dots, w_m\}$ neurons (reference vectors) set. In order to incorporate the SOM neighbourhood relations into the clustering problem, our first modification concerns the construction of this non-complete graph that will be presented to the *Largest First (LF)* algorithm of *Welsh and Powell* [18]. This *non complete edge-weighted graph* is now given by $G_{>\theta,\alpha} = (V, E_{>\theta,\alpha})$, where $E_{>\theta,\alpha}$ is the edge set, where for each two vertices v_i and v_j in V (corresponding to (w_i, w_j) in the map), the edge $(v_i, v_j) \in E_{>\theta,\alpha}$ if $\mathcal{D}(w_i, w_j) > \theta$ or $\mathcal{SN}(w_i, w_j) > \alpha$ where \mathcal{D} is the dissimilarity matrix, \mathcal{SN} is the *SOM rectangular - neighbourhood order matrix*, θ is the *dissimilarity threshold* and α is the *SOM rectangular-neighbourhood order threshold*. This proposal offers the possibility to perform the *minimal coloring based clustering approach* multiple runs, each of them increasing the value of α .

The neurons to be clustered are now depicted by a non-complete edge-weighted graph $G_{>\theta,\alpha}$. Additional modifications of the *minimal coloring based clustering approach* are considered in order to improve the quality of clustering by incorporating the topological relations offered by SOM. The changes concern now the *LF* algorithm used for coloring the graph $G_{>\theta,\alpha}$. In fact, after sorting the vertices of $G_{>\theta,\alpha}$ by decreasing degree, the *LF algorithm* starts from the vertex of V which has the maximum degree Δ (the maximum number of edges from V). The algorithm colours this vertex using the color one. Then, it tries to color the remaining vertices (by respecting the decreasing order of their degree) according to the following principle: each vertex is placed in the first color class for which it has no neighbours. If no such color exists, then a new color is created for it. In the end of the algorithm, each neuron will be colored and neurons having the same color, belong to the same class. Theoretical and practical details can be found in [6].

4 Characterizing McSOM: G-Select

In the previous section, another way for clustering SOM using a graph coloring based approach is presented. Once clustering done, a data partition with an optimal number of classes is obtained. In this section, we introduce a new graph base approach, called G-Select (for Graph based feature selection), to find the local feature subsets that characterize each class.

4.1 Determining “Pivot” Features

The first step of G-Select aims to define the statistical test which allows to determine the “Pivot” features. A “Pivot” feature is the most relevant one in

a class, i.e. it is the feature which contributes most in grouping observations in the associated class.

The statistical test for a given feature in a given class defines a test value (T) which is simple but practically very important [14].

Let be a sample of size n , a class K found by McSOM with cardinality n_K ($n_K < n$). The test value of a feature j in K can be defined by:

$$T_K(j) = \frac{\mu_{jK} - \mu_j}{\sqrt{\frac{n-n_K}{n-1} \times \frac{\sigma_j^2}{n_K}}} \quad (2)$$

Where u_j is the mean value of j in the global sample, its variance is σ_j^2 and its mean in the class is μ_{jK} .

T can be considered as a comparison test between means of different sets. It is classically used to measure dissimilarity between the overall distribution of a feature and that observed in the considered class. We note that if all features participate in the construction of the partition, as in our case, it is not appropriate to define a hypothetic threshold for selecting the most important ones according to T since this value quantifies the relevance of each feature individually and independently of the other ones [14]. Thus, T will be use just for ranking theses features and determining the best one which will serve as a "Pivot" in the construction of tree in the next section. Subsequently, a "Pivot" feature p^* of one class K is defined as follow:

$$p_K^* = \arg \max_{1 \leq j \leq p} |T_K(j)| \quad (3)$$

4.2 Maximum Spanning Tree for Characterization

In this section, we show how to automatically detect the set of features which have a strong multiple correlation and are directly linked to the "Pivot" feature. For that, we propose to perform a maximum spanning tree based method. This technique requires a matrix of weights between vertices (features in our case). We calculate, thus, a matrix of correlations for each class K , where a correlation between two features j and j' is defined in:

$$\text{corr}(j, j') = \frac{1}{\sigma_j \sigma_{j'}} \times \frac{1}{n_K} \sum_{k=1}^{n_K} (z_k^j - \mu_{jK})(z_k^{j'} - \mu_{j'K}) \quad (4)$$

For each K , we can define a weighted graph $G_K(D, CORR)$ is defined, where D is the set of all features (vertices) and $CORR$ is the set of edges evaluated according to [4].

A maximum spanning tree G'_K is a connected and acyclic sub-graph of G_K , for which the sum of edge weights is maximum. This graph allows to detect the features which are directly linked to the "Pivot" feature without using any threshold. Fig. 2

On the left side of Fig. 2, G represents a complet graph where all features are weighted using the correlation values. On the right, we can see the maximum

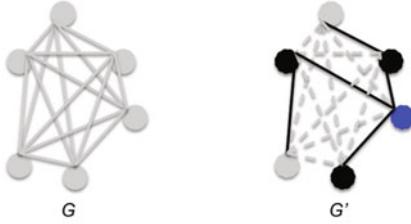


Fig. 2. G : Original Graph; G' :Maximum spanning tree

spanning tree G' obtained from G where are showed: the pivot feature (blue vertex), the features linked to the pivot features (dark vertices) and the edges which contribute to the construction of the tree (solid lines).

For constructing G'_K of a class K , we use the optimized algorithm of Prim [4]:

Let be V and E two empty sets. First, we affect to V one feature from D . The goal is to find the edge (j, j') of $\overline{V} \times V$ having the maximum weight ($\overline{V} = (D - V)$) and to update j in V and (j, j') in E .

The algorithm proceeds as follow:

Let be j_k a selected feature from \overline{V} at step k and assigned to V . We note $V^* = V \cup \{j_k\}$ and $\overline{V}^* = \overline{V} - \{j_k\}$. Thus, at step $k + 1$ we seek :

$$\max_{(j, j') \in \overline{V}^* V^*} \text{corr}(j, j') = \max_{j \in \overline{V}^*} \max_{j' \in V^*} \text{corr}(j, j') \quad (5)$$

A marking procedure consists, in step k , after selecting j_k , to keep for each $j \in \overline{V}^*$ the values $\max_{j' \in V^*} \text{corr}(j, j')$. These values are calculated by the following updating formula :

$$\max_{j' \in V^*} \text{corr}(j, j') = \max_{j' \in V} (\max_{j \in \overline{V}^*} \text{corr}(j, j') \text{corr}(j_k, j)) \quad (6)$$

Then, in step k we memorize for each $j \in \overline{V}^*$ the following values:

- $\text{Pred}(j)$, the farthest vertex of j in V^* :

$$\text{Pred}(j) = \arg \max_{j' \in V^*} \text{corr}(j, j') \quad (7)$$

- $L(j)$, the length between j and $\text{Pred}(j)$:

$$L(j) = \max_{j' \in V^*} \text{corr}(j, j') \quad (8)$$

Consequently, the construction of the tree is done from two vectors : Pred and L of dimension p and from the correlation matrix $CORR$ of dimension $p \times p$. After $(p - 1)$ iterations, the $(p - 1)$ edges are $(j, \text{Pred}(j))$.

Finally, for extracting the features which characterize the class K , we detect the "Pivot" feature p^* according to [3] and we select then its directly linked features from the obtained spanning tree.

5 Results

We performed several experiments on four known problems from UCI Repository of machine learning databases [1]. They are voluntarily chosen (Table I) for comparing our approach (G-Select) with another characterizing clustering methods.

Table 1. Characteristics of used databases

Data sets	N	p	#labels
Wave	5000	40	3
Madelon	2000	500	2
Wdbc	569	30	2
Spamb	4601	57	2

Remark that the used UCI data sets include class information (labels) for each observation. These labels are available for evaluation purposes but not visible to the clustering algorithm. Remember that the objective was to perform unsupervised classification that correctly identifies the underlying classes when the number of clusters is not predefined.

In general, the result of clustering is usually assessed on the basis of some external knowledge about how clusters should be structured. The only way to assess the usefulness of a clustering result is indirect validation, whereby clusters are applied to the solution of a problem and the correctness is evaluated against objective external knowledge. This procedure is defined by [11] as "validating clustering by extrinsic classification" and has been followed in many other studies. Thus, two statistical-matching schemes called Purity and Rand index [10] are used for the clustering accuracy.

5.1 Clustering Characterization

First, we give some details over the values assigned to different parameters for all used databases. For the construction of the maps, we have used an heuristic proposed by Kohonen [13] for automatically providing the initial number of neurons and the dimensions ($nrows \times ncolumns$). Thus, for *Wave*, *Dimension* = 26×14 , for *Wdbc*, *Dimension* = 30×4 , for *Madelon*, *Dimension* = 17×13 and for *Spamebase*, *Dimension* = 38×9 . Then, we remind that the Euclidian distance is applied to define the dissimilarity level D between two p -dimensional referent vectors in the map. Moreover, for each *SOM rectangular-neighborhood order* threshold α (chosen between 1 and the number of SOM' rows), McSOM is performed multiple runs, each of them increasing the value of the dissimilarity threshold θ . Once all neighborhood and dissimilarity threshold values passed, the algorithm provides the optimal partitioning which maximizes *Generalized Dunn's* ($Dunn_G$) quality index [12]. This index is designed to offer a compromise between the inter-cluster separation and the intra-cluster cohesion. So, it is more appropriate to partition data set in compact and well-separated clusters.

$$Dunn_G = \frac{\min_{i,j,i \neq j} d_a(C_i, C_j)}{\max_h s_a(C_h)} \quad (9)$$

- $s_a(C_i)$ is the *average distance* within the cluster C_i
- $d_a(C_i, C_j)$ is the *between-cluster separation*

We start now by analyzing *Wave* data set because we know a priori that it is composed of 21 relevant features (the first ones) and 19 noisy features. The aim is to see if G-Select is able to select just the relevant ones. We have presented this database with all 40 features to McSOM and we have automatically obtained 3 classes with $\langle Purity, Rand \rangle = \langle 0.55, 0.6706 \rangle$. The number of classes corresponds exactly to the real defined one. Then, we have applied G-Select over these 3 classes and we have obtained for each one its characterizing features. Thus, G-Select provided for each class its “Pivot” feature which is directly linked to a subset of features extracted from the maximum spanning tree over all features. So, [14, *15*, 16, 17, 19] are selected for the first class, [2, 6, *7*, 8, 30] for the second class and [1, 9, 10, *11*, 12] for the third one (the numbers between ** represent ”Pivot” features). We can see that G-Select allows the selection of 14 relevant features (with one noisy feature: 30) for the characterization. Nevertheless, The approach provides good rates on *Purity* and *Rand* with 0.5680 and 0.6713, respectively. These rates are better than those found by another weighting based characterization approach (lwd-SOM: $\langle 0.5374, 0.6068 \rangle$, lwo-SOM: $\langle 0.5416, 0.6164 \rangle$) in [8] with a minimum number of features (Table 2).

Table 2. G-Select vs lwo-SOM and lwo-SOM over *Wave* database

	lwd-SOM	lwo-SOM	G-Select
Classes:[features]	$cl_1:[6-15]$ $cl_2:[4-10]$ $cl_3:[7-19]$	$cl_1:[3-8,11-16]$ $cl_2:[8-11,14-19]$ $cl_3:[3-20]$	$cl_1:[14-17,19]$ $cl_2:[2,6-8,30]$ $cl_3:[1,9-12]$
Purity	0.5374	0.5416	0.5680
Rand	0.6068	0.6164	0.6713

We provide in Table 3 a comparison of the quality of McSOM before and after feature selection by G-Select over the other databases. In Table 4 we present the results of our approach versus lwd-SOM and lwo-SOM over the same databases.

We can see from all these tables that G-Select provides:

- An automatic characterization of classes from clustering, with the selection of an optimal number of features,
- An important elimination of noise,
- An improvement of SOM clustering after feature selection
- A good rates on *Purity* and *Rand* after selection compared to another characterization based methods.

Table 3. Clustering accuracy before and after feature selection

Data sets	Before Selection		After Selection	
	Rand	Purity	Rand	Purity
Wdbc	0.6769	0.7979	0.8843	0.9385
Madelon	0.5085	0.5825	0.5094	0.5945
Spambase	0.5195	0.6059	0.5197	0.6624

Table 4. Local feature selection for each class. [features]; $\langle Purity \rangle$; [*]: [49 65 106 129 242 339 344 356 443 454 476 494].

Data sets	# cl	lwd-SOM	lwo-SOM	G-Select
Wdbc	2	$cl_1 - cl_9 : [4,24]$ $\langle 0.6274 \rangle$	$cl_1 - cl_9 : [4,24]$ $\langle 0.8682 \rangle$	$cl_1:[7,8,23,28]$ $cl_2:[3,7,8,28]$ $\langle 0.9285 \rangle$
Madelon	2	$cl_1:1; cl_2: [91, 281, 403-424]$ $\langle 0.5242 \rangle$	$cl_1:1; cl_2:[242, 417-452]$ $\langle 0.5347 \rangle$	$cl_1 - cl_4: [*]$ $\langle 0.5945 \rangle$
Spamb	2	$cl_1:56 ; cl_2:57$ $\langle 0.6103 \rangle$	$cl_1:56 ; cl_2:57$ $\langle 0.6413 \rangle$	$cl_1:[26,32,34,55]$ $cl_2:[27,29,31,32, 34,40,43]$ $\langle 0.6624 \rangle$

6 Conclusion and Future Work

We proposed in this paper a graph based framework for clustering and characterizing Self-organizing map. The proposal concerns more particularly G-Select which provides local feature selection from clustering obtained by a recently developed method called McSOM. in this approach we combined a statistical test for detecting "Pivot" features from obtained classes, with maximum spanning tree for extracting subsets of relevant features allowing their characterization. Some interesting issues can be raised from this work, for example the construction of graphs from several "Pivot" features and the optimization of the proposed approach to extend it for large databases analysis.

References

1. Asuncion, A., Newman, D.: UCI machinelearning repository (2007), <http://www.ics.uci.edu/mlearn/MLRepository.html>
2. Benabdeslem, K., Lebbah, M.: Feature selection for self organizing map. In: IMAC/IEEE ITI, pp. 45–50 (2007)
3. Cabanes, G., Bennani, Y.: A simultaneous two-level clustering algorithm for automatic model selection. In: ICMLA, pp. 316–321 (2007)
4. Cormen, T.H., Leiserson, E.C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT Press and McGraw-Hill (2001)

5. Dash, M., Choi, K., Scheuermann, P., Liu, H.: Feature selection for clustering-A filter solution. In: Proceedings of the IEEE International Conf. on Data Mining, pp. 115–122 (2002)
6. Elghazel, H., Benabdeslem, K., Kheddouci, H.: McSOM: Minimal coloring of self organizing map. In: Huang, R., Yang, Q., Pei, J., Gama, J., Meng, X., Li, X. (eds.) Advanced Data Mining and Applications. LNCS (LNAI), vol. 5678, pp. 128–139. Springer, Heidelberg (2009)
7. Frigui, H., Nasraoui, O.: Unsupervised learning of prototypes and attribute weights. *Pattern Recognition* 37(3), 567–581 (2004)
8. Grozavu, N., Bennani, Y., Lebbah, M.: From feature weighting to cluster characterization in topographic unsupervised learning. In: IEEE International Joint Conference on Neural Network, pp. 1005–1010 (2009)
9. Huang, J.Z., Ng, M.K., Rong, H., Li, Z.: Automated feature weighting. *IEEE Trans. Pattern Anal. Mach. Intell.* 27, 657–668 (2005)
10. Hubert, L., Arabie, P.: Comparing partitions. *Journal of Classification* 2, 193–218 (1985)
11. Jain, A., Murty, M.: Data clustering: A review. *ACM Computing Surveys* 31, 264–323 (1999)
12. Kalyani, M., Sushmita, M.: Clustering and its validation in a symbolic framework. *Pattern Recognition Letters* 24(14), 2367–2376 (2003)
13. Kohonen, T.: Self organizing Map. Springer, Berlin (2001)
14. Lebart, L., Morineau, A., Warwick, K.: Multivariate descriptive statistical analysis. John Wiley and Sons, New York (1984)
15. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: 5-th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
16. Rand, W.M.: Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association* 66, 846–850 (1971)
17. Vesanto, J., Alhoniemi, E.: Clustering of the self organizing map. *IEEE Transactions on Neural Networks* 11(3), 586–600 (2000)
18. Welsh, D.J.A., Powell, M.B.: An upper bound for the chromatic number of a graph and its application to timetabling problems. *Computer Journal* 10(1), 85–87 (1967)

Clustering Using Elements of Information Theory

Daniel de Araújo^{1,2}, Adrião Dória Neto², Jorge Melo², and Allan Martins²

¹ Federal Rural University of Semi-Árido, Campus Angicos,
Angicos/RN, Brasil
daniel@ufersa.edu.br

² Federal University of Rio Grande do Norte,
Departament of Computer Engineering and Automation, Natal/RN, Brasil
{adriao,jdmelo,allan}@dca.ufrn.br

Abstract. This paper proposes an algorithm for clustering using an information-theoretic based criterion. The cross entropy between elements in different clusters is used as a measure of quality of the partition. The proposed algorithm uses “classical” clustering algorithms to initialize some small regions (auxiliary clusters) that will be merged to construct the final clusters. The algorithm was tested using several databases with different spatial distributions.

Keywords: Clustering, Cluster Analysis, Information Theoretic Learning, Complex Datasets, Entropy.

1 Introduction

There are many fields that clustering techniques can be applied such as marketing, biology, pattern recognition, image segmentation and text processing. Clustering algorithms attempt to organize unlabeled data points into clusters in a way that samples within a cluster are “more similar” than samples in different clusters [1]. To achieve this task, several algorithms were developed using different heuristics. Although in most part of clustering tasks no information about the underlying structure of the data is used during the clustering process, the majority of clustering algorithms requires the number of classes as a parameter to be given *a priori*. Moreover, the spatial distribution of the data is another problematic issue in clustering tasks, since most part of the algorithms have some bias to a specific cluster shape. For example, single linkage hierarchical algorithms are sensitive to noise and outliers tending to produce elongated clusters and k -means algorithm yields to elliptical clusters.

The incorporation of spatial statistics of the data gives a good measure of spatial distribution of the objects in a dataset. One way of doing that is use information-theoretic elements to help the clustering process. More precisely, Information Theory involves the quantification of information in a dataset using some statistical measures. Recently, [2][3][4] achieved good results using some elements of information theory to help clustering tasks. Based on that, this

paper proposes a information-theoretic based heuristic to clustering datasets. In fact, we propose a iterative two-step algorithm that tries to find the best label configuration by switching the labels according to a cost function based on the cross entropy [3]. The utilization of statisticial based measures enables the algorithm to cluster spatial complex datasets.

The paper is organized as follows: in Sect. 2 we make some consideration about the information theory elements used in the clustering algorithm; in Sect. 3 we describe the information-theoretic based clustering criterion; in Sect. 4 we present the proposed clustering algorithm; Sect. 5 shows some results obtained and in Sect. 6 the conclusions and considerations are made.

2 Information Theoretic Learning

Information Theory involves the quantification of information in a dataset using some statistical measures. The most used information-theoretic measures are Entropy and its variation. Entropy is a measure of uncertainty about a stochastic event or, alternatively, it measures the amount of missing information associated with an event [5]. From the idea of entropy arose other measures of information, like Mutual information [4], Kullback-Leibler divergency [6], cross entropy [4] and joint entropy [4].

Let us consider a dataset $X = \{x_1, x_2, \dots, x_n\} \in R^d$ with independent and identically distributed (iid) samples. The most traditional measure of information is the Shannon's entropy H_s , that is given by [7]:

$$H_s(\mathbf{x}) = \sum_{k=1}^n p_k I(p_k) \quad (1)$$

where $\sum_{k=1}^n p_k = 1$ and $p_k \geq 0$.

After that, Alfred Renyi proposed another measure of entropy in the 60's, known as Renyi's entropy [8]:

$$H_R(\mathbf{x}) = \frac{1}{1-\alpha} \ln \int f^\alpha(\mathbf{x}) d\mathbf{x} \quad \alpha \geq 0, \alpha \neq 1 \quad (2)$$

The most used variation of the Renyi entropy is its quadratic form, where $\alpha = 2$:

$$H_R(\mathbf{x}) = -\ln \int f_x(\mathbf{x})^2 dx \quad (3)$$

In (3), $f_x(\mathbf{x})$ is a probability density function (PDF). So , it is necessary the estimation of that PDF and as we are working in a clustering context task, we don't have any information about the underlying structure of the data. Then, we used one of the most popular approach to make nonparametric estimation: the Parzen Window estimator [9]. The Parzen Window can be written as:

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N G(\mathbf{x} - \mathbf{x}_i, \sigma^2) \quad (4)$$

where $G(\mathbf{x}, \sigma^2)$ is multivariate Gaussian function defined as:

$$G(\mathbf{x}, \sigma^2) = \frac{1}{(2\pi)^{d/2} |\sum|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu) | \sum |^{-1} (\mathbf{x} - \mu)^T\right) \quad (5)$$

in this case, \sum is the covariance matrix and d is the dimension of \mathbf{x} . When we substitute (4) and (5) in (3), we have:

$$\begin{aligned} H_R(\mathbf{x}) &= -\ln \int \left(\frac{1}{N} \sum_{i=1}^N G(\mathbf{x} - \mathbf{x}_i, \sigma^2) \right) \left(\frac{1}{N} \sum_{j=1}^N G(\mathbf{x} - \mathbf{x}_j, \sigma^2) \right) \\ &= -\log \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G(\mathbf{x}_i - \mathbf{x}_j, 2\sigma^2) \end{aligned} \quad (6)$$

According to [4] this equation is known as Information Potential, because the similarity to potential energy between physical particles. The Information Potential was successfully used in several works as distance measure or clustering criterion [3/2/10].

As we can notice, entropy measures the information of one random variable. When we are interested in quantify the interaction between two different datasets, one choice is to compute the cross entropy between them [3]. Extending the concepts of Renyi's entropy, we can define formally the cross entropy between two random variables $X = (\mathbf{x}_i)_{i=1}^N$ and $Y = (\mathbf{y}_j)_{j=1}^M$ as:

$$\begin{aligned} H(X; Y) &= -\log \left(\int p_{\mathbf{X}}(t)p_{\mathbf{Y}}(t)dt \right) \\ &= -\log \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M G(\mathbf{x}_i - \mathbf{y}_j, 2\sigma^2) \end{aligned} \quad (7)$$

The cross entropy criterion is very general and can be used either in a supervised or unsupervised learning framework. We are using an information-theoretic criteria based on maximization of cross entropy between clusters.

3 The Proposed Clustering Criterion

Two major issues in clustering are: how to measure similarity (or dissimilarity) between objects or clusters in the dataset and the criterion function to be optimized [1]. For the first issue, the most obvious solution is use the distance between the samples. If the distance is used, then one would expect the distance between samples in the same cluster to be significantly less than the distance between samples in different clusters [1].

The most common class of distance used in clustering tasks is the Euclidean distance [1][11][12]. Clusters formed using this kind of measure are invariant to translation and rotation in feature space. Some applications, like gene expression analysis, rather use correlation or association coefficients to measure similarity between objects [13][12].

About the criterion function, one of the most used criterion is the sum of squared error. Clusterings of this type produce a partition containing clusters with minimum variance. However, the sum of squared error is most indicated when the natural clusters form compact and separated clouds [1][14]. Another usual class of algorithms are the agglomerative hierarchical algorithms. That class of algorithm represents the dataset as a hierarchical tree where the root of the tree consists of one cluster containing all objects and the leaves are singleton clusters [1]. The spatial shape of the partitions produced by hierarchical clustering algorithms depends of the linkage criterion used. Single linkage algorithms are sensitive to noise and outliers. Average and complete linkage produces elliptical clusters.

This paper proposes the use of cross entropy as a cost function to define the clusters of a given dataset. The objective of the algorithm is to maximize cross entropy between all clusters. As pointed earlier, the cross entropy is based on a entropy measure that need the estimation of the data density distribution. So, the approach utilized in this work is the cross entropy using Parzen Window estimation method described in Sect 2.

Using elements of information theory as clustering criterion takes advantage of the underlying statistical information that the data carries. Indeed, the algorithm makes no assumption of the statistical distribution of the data, instead of that, it tries to estimate that distribution and uses it as a measure of similarity between clusters.

When the cross entropy is utilized in the clustering context, it is taken into account the relation between each group. This relation is showed as the influence that one cluster can have on another.

4 The Proposed Clustering Algorithm

The main goal of a clustering algorithm is to group objects in the dataset putting into the same cluster samples that are similar according to a specific clustering criterion.

Iterative distance-based algorithms form a effective class of techniques to deal with clustering tasks. They work based on the minimization of the total squared distance of the samples to their cluster centers. Although, they have some problematic issues like the sensibility to the initial guesses of the cluster centers and restrictions related to the spatial distribution of the natural groups [11].

One way of using iterative distance-based algorithms efficiently is to cluster the dataset using a high number of clusters, i.e., using more clusters than the intended number of clusters in final partition and after that merge those clusters to

form larger and more homogeneous clusters. Many authors apply this approach in their works and had good results for spatial complex datasets [32][10].

We are using cross entropy as a cost function and its calculation utilizes all data points of each cluster when we use the Parzen Window estimator. So, the larger the cluster is, the longer is the time to compute the cross entropy. When we apply the strategy of split the dataset into several small regions we treat two issues: the small regions are usually more homogeneous and easier to cluster using algorithms that could not correctly cluster the entire dataset, like k -means and with smaller clusters the time consumption to compute cross entropy decreases.

Based on that, the proposed clustering algorithm works in a iterative two-step procedure: first, the dataset is divided in a large number of compact small regions, named auxiliary clusters. Then, each region is randomly labeled according to the specified number of cluster but not yet corresponding to the final partition labels, e.g., if we are dealing with a two-class dataset, two kinds of labels will be distributed to the auxiliary clusters and the two clusters are composed by all regions sharing the same labels. The second step of the algorithm consists in switch the labels of each small region checking whether this change increases the cost function. Every time the cost function is increased the change that causes the raise is kept, otherwise it is reversed. This processes is repeated until there is no changes in the labels.

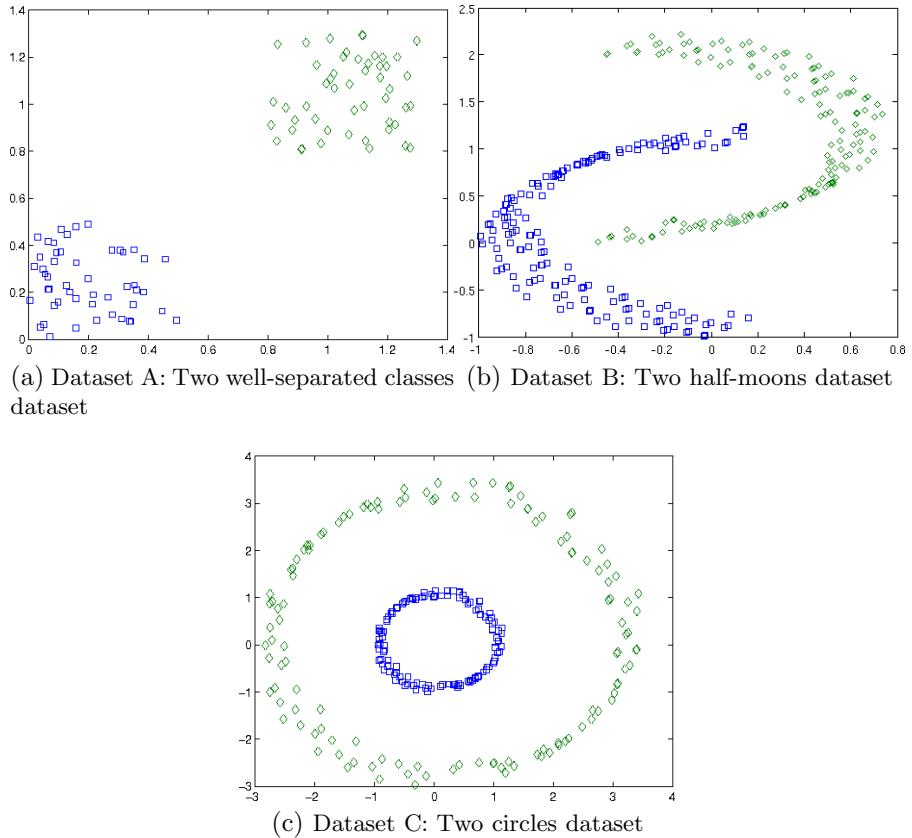
The small regions found in the initial phase work as auxiliary clusters that will be used to discover final clusters in the dataset. The task of finding auxiliary clusters can be made by a “traditional” clustering algorithm like k -means, competitive neural networks or hierarchical algorithms. In our case, we used the k -means algorithm, that is a well-known clustering algorithm [11].

The label switch process take each auxiliary cluster and changes its label to a different one. Then, the cost function is calculated and if some increase is noticed, that change is recorded and the new label configuration is assumed. After all clusters labels have been changed, the process starts again and continues until there is no new changes in any auxiliary cluster label. This can be seen as a search for the optimal label configuration of the auxiliary clusters and for consequence the optimal configuration of the clusters in the final partition.

Due to the initial random assigning process, the proposed algorithm is non-deterministic and can produce different results for different initialization. Also, the number of auxiliary clusters direct influences the overall performance.

5 Experimental Results

To test the performance of the proposed clustering algorithm we used some traditional clustering datasets with simple and complex spatial distributions. Figure 1 illustrates all datasets. Notice that the dataset A (Fig. 1a) is a classic two well-separated clouds classes and it is the simplest dataset of all used in this paper. The dataset B (Fig. 1b) and dataset C (Fig. 1c) have a more complex spatial distribution.

**Fig. 1.** Datasets

If we use the same traditional center-based technique (k -means) that we used to create the initial auxiliary clusters of our algorithm, it would be able to correctly separate the clusters of only one dataset, the simplest one (dataset A). This happens because the clusters into that dataset have spherical shapes. For the rest of tested datasets, k -means could not achieve good results. Figure 2 show the performance of k -means over all datasets tested with our proposed algorithm.

As pointed out earlier, the number of auxiliary clusters and the number of final clusters are needed to start the process. Running some pre-experimental tests, always with the number of clusters equals to the real number of classes, we could find which number of auxiliary cluster is more suitable based on the cross entropy value. Also, since there is some randomness in the initial label assigning process, we run 10 simulations using each dataset and show here the one with greater cross entropy.

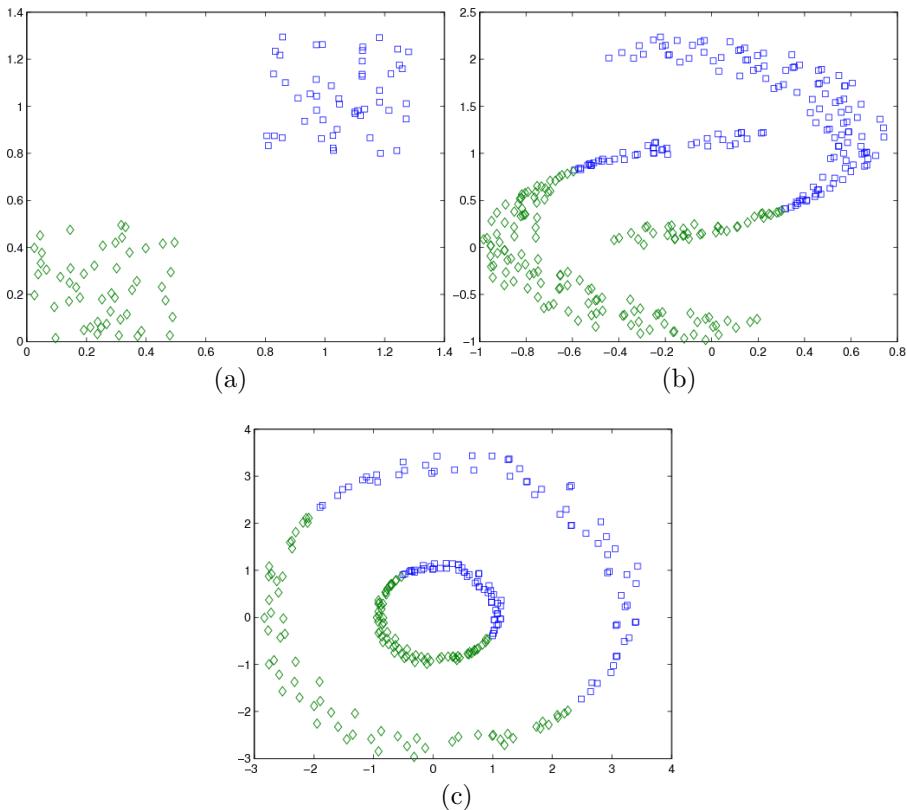


Fig. 2. *k*-means clustering for all datasets

The results achieved using the proposed algorithm are shown in Figs. 3, 4 and 5. For each dataset it is shown the entire process of clustering. In each Fig., the first picture shows the dataset clustered using the auxiliary clusters. The second picture illustrates the initial labels assigned randomly to each auxiliary clusters. The rest of pictures composes the switching label process leading to the last and final picture represent the final partition.

As we can see, the algorithm performed the correct separation of the classes in all cases. The dataset A could be correctly clustered using any number of auxiliary clusters. The other datasets, despite the spatial complexity, could be correctly clustered using the specified parameters.

Those results are in agreement with other works using Information Theory to help the clustering process. For instance, [10] used Renyi's entropy as a clustering criteria, [15] proposed a co-clustering algorithm using mutual information and [2] who developed a clustering algorithm based on Kullback-Leibler divergence.

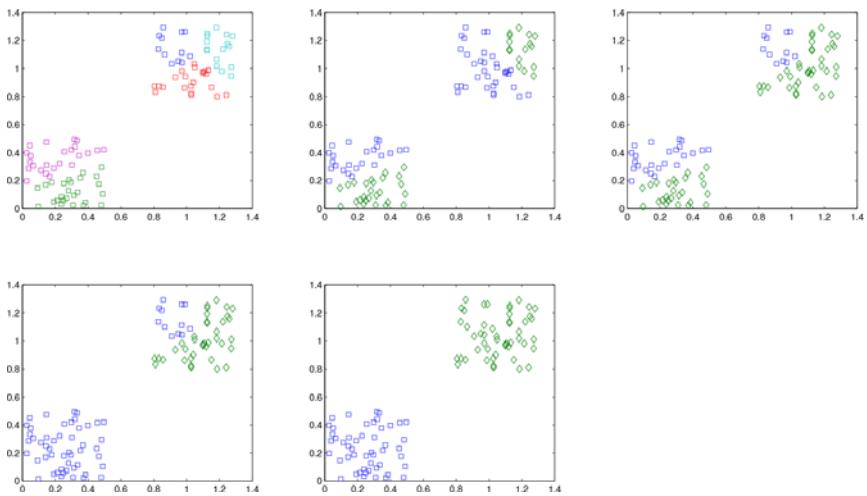


Fig. 3. Partition achieved using five auxiliary clusters

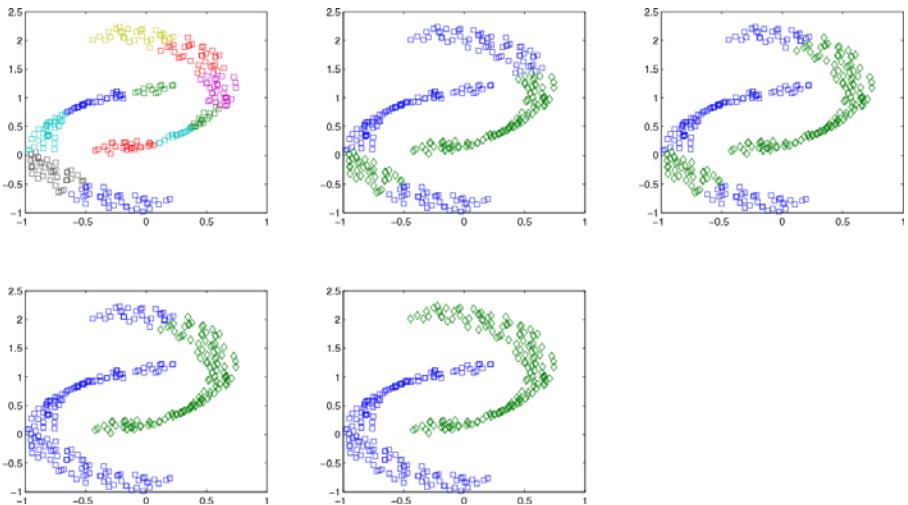


Fig. 4. Partition achieved using 11 auxiliary clusters

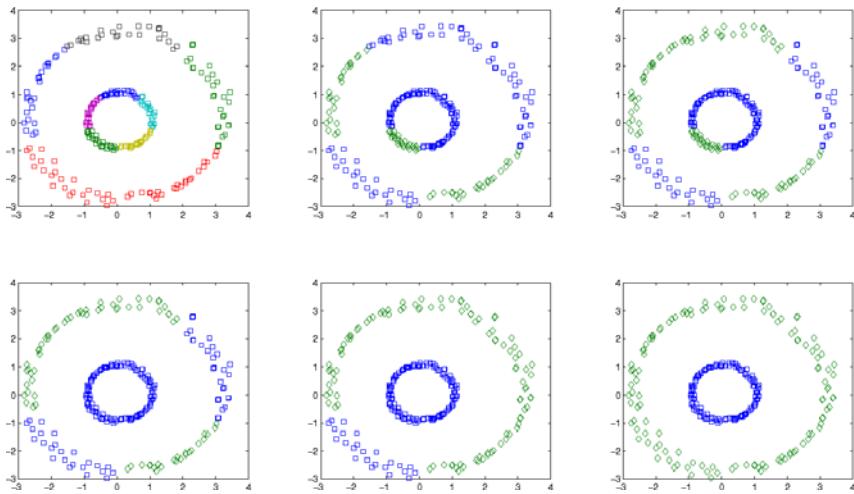


Fig. 5. Partition achieved using 10 auxiliary clusters

6 Conclusions

In this paper we proposed a clustering algorithm using elements of information theory as a cost function criterion. A two-step heuristic creates a iterative procedure to form clusters. We also tested the algorithm using simple and complex spatial distribution. When the correct number of auxiliary clusters is used, the algorithm performed perfectly.

The use of statistical based measures enables the algorithm to cluster spatial complex datasets using the underlying structure of the data. But it is reasonable to think that the algorithm has some issues derived from its base structure based on the k -means clustering algorithm and the kernel function used to estimate the density probability. But, considering the initial experimental tests, which achieved good results, there is a lot of variables that can be changed to improve the capacity of the algorithm to some clustering tasks.

References

1. Duda, R., Hart, P., Stork, D.: *Pattern Classification*. Wiley, Chichester (2001)
2. Martins, A.M., Neto, A.D.D., Costa, J.D., Costa, J.A.F.: Clustering using neural networks and kullback-leibler divergency. In: Proc. of IEEE International Joint Conference on Neural Networks, vol. 4, pp. 2813–2817 (2004)
3. Rao, S., de Medeiros Martins, A., Príncipe, J.C.: Mean shift: An information theoretic perspective. *Pattern Recogn. Lett.* 30(3), 222–230 (2009)
4. Príncipe, J.C.: *Information theoretic learning*, vol. 7. John Wiley, Chichester (2000)
5. Príncipe, J.C., Xu, D.: Information-theoretic learning using renyi's quadratic entropy. In: Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation, Aussois, pp. 407–412 (1999)

6. Kullback, S., Leibler, R.A.: On information and sufficiency. *The Annals of Mathematical Statistics* 22(1), 79–86 (1951)
7. Shannon, C.E.: A mathematical theory of communication. *Bell System Technical Journal* 27, 379–423, 625–656 (1948)
8. Cover, T.M., Thomas, J.A.: *Elements of Information Theory*, 2nd edn. John Wiley, Chichester (1991)
9. Parzen, E.: On the estimation of a probability density function and the mode. *Annals of Mathematical Statistics* (33), 1065–1076 (1962)
10. Gokcay, E., Principe, J.C.: Information theoretic clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* 24(2), 158–171 (2002)
11. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. Morgan Kaufmann, San Francisco (2005)
12. Hair, J.F. (ed.): *Multivariate data analysis*, 6th edn. Pearson/Prentice Hall, Upper Saddle River, NJ (2006)
13. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., Lander, E.S.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 286(5439), 531–537 (1999)
14. Jain, A.K., Dubes, R.C.: *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River (1988)
15. Dhillon, I., Mallela, S., Modha, D.: Information-theoretic co-clustering. In: Grnwald, P. (ed.) ACM SIGKDD, The Minimum Description Length Principle (2003)

A Path Planning Method for Human Tracking Agents Using Variable-Term Prediction

Noriko Takemura¹, Yutaka Nakamura¹, Yoshio Matsumoto², and Hiroshi Ishiguro¹

¹ Osaka University

² Advanced Industrial Science and Technology, Japan

{takemura.noriko,nakamura}@is.sys.es.osaka-u.ac.jp

Abstract. This paper deals with a multi-agent path planning problem for tracking humans. Path of agents is planned based on the similarity between the prediction of the intensity of humans existing and the intensity of field of view of agents. Since the prediction is not always accurate, we proposed the an path planning method method where prediction length is varied based on the reliability of the prediction. We conducted computer simulation and results showed that our path planning method works well even under changing circumstances.

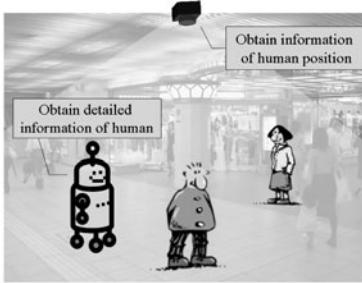
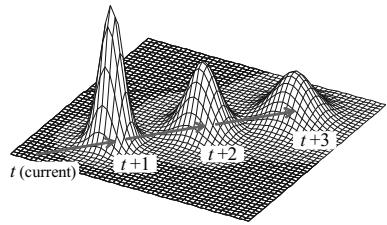
Keywords: path planning, multiple human tracking, variable term prediction.

1 Introduction

Visual surveillance methods have been intensively studied in order to ensure the security. For a surveillance system, not only information of human position but also detail information such as human behaviors and their characteristics are important. In this research, we focus on a path planning for mobile agents to acquire detail information of tracked humans. In our method, the path is determined based on the long-term prediction of human positions. Some methods with long-term prediction have been proposed [1] and it has been confirmed that an accurate prediction of human position improves performance of human tracking [2] [5]. Since accurate long-term predictions are not always possible, we propose a path planning method with variable-term prediction. In this research, prediction length is determined according to the consistency of prediction which estimates the reliability of prediction. We conduct simulation experiments and experimental results show our method works well even in a changing environment.

2 Human Tracking Task

The purpose of our path planning is to make agents to be close to humans in order to obtain their detail behaviors. We assume that the current position and velocity of each human can be measured by sensors embedded in the environment(Fig 1[4][3]). We also assume that no occlusions between humans and agents occur. Since the closer a distance between a human and an agent, the more detail of the human behavior might be obtained, a set of planned paths of agents is evaluated based on following criteria NND(Nearest Neighbor Distance) : $NND = \min_j \|\bar{x}_{h,i}(t) - \bar{x}_{c,j}(t)\|$, where $\bar{x}_{h,i}$ and $\bar{x}_{c,j}$ are the positions of the i -th human and the j -th agent. The aim of this research is to develop a path planning method for agents which minimizes this criteria.

**Fig. 1.** Human tracking problem**Fig. 2.** Prediction model of human movement

3 Path Planning Procedure

In our method, the number of prediction steps $T(t)$ is determined according to consistency of prediction of humans movements at the beginning of each time step, and then, human movements are predicted during $T(t)$ steps. After that, agents paths during $T(t)$ steps are planned based on the prediction.

Prediction Model of Human Movement. Human movements are predicted by uniform linear motion model considering uncertainty (Fig. 2). The intensity of human existing at time $t + \tau$ predicted at time t is assumed to be calculated as $H(\mathbf{x}, t + \tau; t) = \sum_i \mathcal{N}(\mathbf{x} | \bar{\mathbf{x}}_{h,i}(t) + \bar{\mathbf{v}}_{h,i}(t)\tau, \sigma_h(t+\tau))$ where $\bar{\mathbf{v}}_{h,i}(t)$ is the velocity of human at current time t and standard deviation is $\sigma_h(t + \tau) = \sigma_h(t)\sqrt{\tau}$.

Determining the Number of Prediction Steps. Although path planning using accurate long-term prediction would be efficient, one using inaccurate long-term prediction might decrease performance of tracking in some case. The prediction of human positions is assumed to be accurate if predicted movements of humans at subsequent time steps are similar, i.e. consistent. The consistency of the prediction is calculated by the KL divergence between intensity of humans existing at time $t + \tau$ predicted at current time t and that predicted at previous time step $t - 1$: $Consistency(t, \tau) = \int_{-\infty}^{\infty} H(\mathbf{x}, t + \tau; t - 1) \log \frac{H(\mathbf{x}, t + \tau; t - 1)}{H(\mathbf{x}, t + \tau; t)} d\mathbf{x}$. The number of prediction steps $T(t)$ is determined based on τ^* which is the largest number where $Consistency(t, \tau)$ is below the threshold.

Path Planning. The intensity of the cameras at time $t + \tau$ planned at time t is defined by mixed Gauss functions : $C(\mathbf{x}, t + \tau; t) = \sum_j \mathcal{N}(\mathbf{x}_{c,j}(t + \tau; t), \sigma_c^2)$, where \mathbf{x}_c is the planned position of each agent. The path is determined based on the KL divergence between H and C , and the path which minimizes the KL divergence is obtained by a gradient method. The procedure of this method is similar to one of a clustering method where each cluster center is corresponding to the position of each agent. Since H and C are Gaussian mixture functions and therefore the analytical calculation is difficult, KL divergence (objective function) is calculated by a sampling method: $F(\mathbf{x}_c, t) = KL(H(\mathbf{x}, t + \tau; t) || C(\mathbf{x}, t + \tau; t)) = \sum_{\tau=1}^T \frac{1}{N_s(\tau)} \sum_{s=1}^{N_s(\tau)} \log \frac{H(\mathbf{x}_s, t + \tau; t)}{C(\mathbf{x}_s, t + \tau; t)}$, where \mathbf{x}_s

is the s -th tentative human position sampled from the existing intensity and $N_s(\tau)$ is the number of sampling points of each human. The positions of agents are determined by minimizing this objective function with considering the maximum velocity of the agent by a barrier method.

Generation of Initial Solution. Since the number of iterations in the gradient method depends on the distance between the initial solution and the suboptimal solution, it seems to be better that the initial solution is generated near by the optimal solution. In our method, the initial solution is generated as $\bar{x}_c^{(0)}(t + \tau; t) = \bar{x}_c(t + \tau; t - 1)$ using the planned paths of the previous time step.

4 Simulation Experiment

We conducted experiments in a stationary environment where human movements do not change and in a dynamic environment where human movements sometimes change.

Tracking in a Stationary Environment. The experiment is conducted in the case that two groups of humans repeat enter the area and move with uniform linear motion regularly (Fig. 3). Humans before entering the path planning area are visible and taken account into the path planning. Fig. 3 shows experimental results under the stationary environment. According to Fig. 3 each pathway of agents depends on prediction length and the agents seem to take two strategies of human tracking (Fig. 3 (S1), (S2)). Under the environment which the prediction can be accurate, the method with long-term prediction is efficient.

Tracking in a Dynamic Environment. The simulation experiment are conducted in the case of human movement model (Mode 1~3) switching dynamically. In Mode 1 ($t = 1 \sim 100$) and Mode 3 ($t = 301 \sim 400$), humans move with uniform linear motion as same as the previous experiment and in Mode 2 ($t = 101 \sim 300$), humans turn $\pm 90^\circ$ randomly. We compare three method: *Constant* (the number of prediction

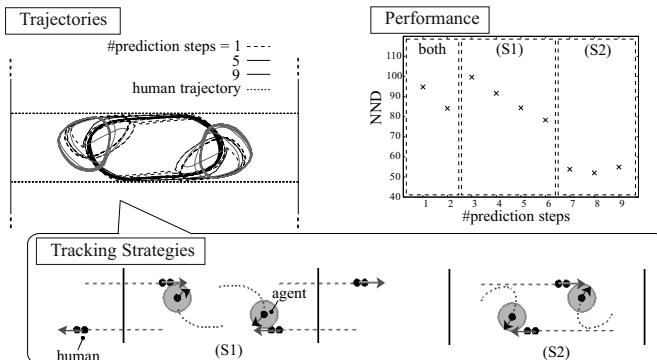
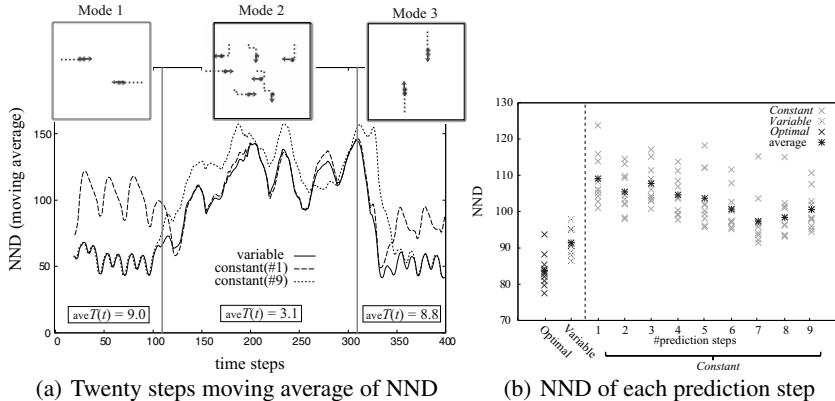


Fig. 3. Experimental results under stationary environment

**Fig. 4.** Experimental results under dynamic environment

steps is fixed at 1, 2, ⋯, 9), *Variable* (proposed method) and *Optimal* (20 steps movements of all humans are given). Fig. 4 shows experimental results of ten runs under the dynamic environment. According to Fig. 4, when the prediction was accurate (Mode 1, 3), the method with a long-term prediction (*Constant*(#9)) is efficient, on the other hand, when the prediction was inaccurate (Mode 2), the method with a short-term prediction (*Constant*(#1)) is efficient. The behavior of the *Variable* was similar to that of *Constant*(#9) during Mode 1 or 3 and was similar to that of *Constant*(#1) during Mode 2. The performance of the *Variable* using each data set was always better than those of *Constants*.

5 Conclusions

In this paper, we proposed a path planning method using long-term prediction of human movements. In our method, the prediction length is varied according to the similarity of predictions of human movements because these prediction seems to be similar when the prediction model is adequate. Experimental results show the performance of the human tracking by our method can be kept high even in a changing environment.

References

1. Bruce, A., Gordon, G.: Better motion prediction for people-tracking. In: Proc. ICRA (2004)
2. Jung, B., Sukhatme, G.: Proc. ICRA, pp. 2189–2195 (2004)
3. Khan, S., Shah, M.: Tracking multiple occluding people by localizing on multiple scene planes. IEEE Trans. of Pattern Anal. and Mach. Intell. (2008)
4. MacDorman, K.F., et al.: IEEE MultiMedia 14-2, 38–49 (2007)
5. Takemura, N., Miura, J.: Journal of the RSJ 25(8), 82–89 (2007)

Three-Layer Feedforward Structures Smoothly Approximating Polynomial Functions

Yuichi Nakamura¹ and Masahiro Nakagawa²

¹ Anan National College of Technology,
265 Aoki, Minobayashi, Anan, Tokushima, 774-0017 Japan
yuichi@anan-nct.ac.jp

² Nagaoka University of Technology,
1603-1 Kamitomioka, Nagaoka, Niigata, 940-2188 Japan
masanaka@vos.nagaokaut.ac.jp

Abstract. This paper considers a structure of three-layer feedforward networks that approximate polynomial functions. The feedforward network has some system parameters such as the coupling coefficients and the biases. The structure of the feedforward network is determined by the system parameters. For any polynomial function, a simple calculation method of the parameters is proposed when the three-layer feedforward network sufficiently approximates the polynomial function. Moreover, it is shown that the obtained feedforward network smoothly approximates the polynomial function.

Keywords: Feedforward network, Structure, Approximation, Polynomial function.

1 Introduction

A feedforward neural network (FNN) defines a mapping of the input-output relation. As the theoretic capability of FNNs, the approximation possibility to several functions has been considered. It has been shown that the input-output mapping of three-layer feedforward neural networks (TLFNNs) can approximately realize any continuous function with an arbitrary accuracy on a compact set [23]. In the above studies, the sigmoid function is used as the activation function of the neurons. In the case of non-polynomial activation function, Leshno *et al.* [5] have proved the existence of TLFNNs which approximate a given function. Moreover, Hornik *et al.* [4] have proved that the derivative of the function can be simultaneously approximated by the TLFNN if the given function is smooth. These results are theoretic and non-constructive approaches, and give no specific information of the structure of FNNs.

There are studies for constructive method of FNNs. Meltser *et al.* [6] have derived a constructive method for approximating in the maximal error norm and verified the appropriate TLFNN architecture for a problem. Suzuki [9] has proved the constructive theorems of TLFNNs approximating periodic Lebesgue integrable functions, and has clarified the number of hidden-layer units, and the

approximation error. Scarselli and Tsoi^[7] have compared some approximation theorems and methods, and have proposed a learning algorithm for FNNs. However, the calculations of these constructive methods are comparatively complex.

Toda-Funahashi-Usui^[10] have demonstrated that a structure of TLFNNs approximating multipliers can be obtained by a simple calculation. Referring to the method, we propose a simple calculation method of the structure of TLFNNs approximating multivariate polynomial functions. The structure of the TLFNN is determined by the system parameters: the coupling coefficients between the neurons, the biases of the neurons. The proposed method is derived by the Taylor expansion of the activation function and by the matrix operation. Moreover, it is shown that the obtained TLFNN smoothly approximate the polynomial function.

2 Main Results

We treat the TLFNNs that are composed of a linear input-layer, a linear output-layer, and a nonlinear middle-layer. This type of TLFNN is applied to several engineering field such as the system control, etc^[18].

Let N^I be the number of the input-layer neurons, N^M be the number of the middle-layer neurons, and N^O be the number of the output-layer neurons. Then, the input-output relation of the TLFNN defines a mapping $g : \mathbb{R}^{N^I} \rightarrow \mathbb{R}^{N^O}$ represented by

$$g(\mathbf{x}) = W^{OM}\sigma(W^{MI}\mathbf{x} + B^M) + B^O, \quad (1)$$

where, $\mathbf{x} = [x_1, \dots, x_{N^I}]^{\text{tr}}$ is the input vector composed of the input values^[1]. W^{MI} and W^{OM} are the coupling coefficients matrices from the input-layer to the middle-layer and from the middle-layer to the output-layer, respectively. B^M and B^O are the biases vectors of the middle-layer and of the output-layer, respectively. $\sigma : \mathbb{R}^{N^M} \rightarrow \mathbb{R}^{N^M}$ is the activation mapping that operates the activation function σ to each component defined by $\sigma(\mathbf{x}) = [\sigma(x_1), \dots, \sigma(x_{N^I})]^{\text{tr}}$. The proposed method requires the following characteristics of σ : (a) σ is continuously differentiable to sufficient degrees, (b) the derivative $\sigma^{(k)}$ is non-constant function for any k . The structure of the TLFNN is decided by calculating the system parameters that are the coupling coefficients matrices W^{MI} , W^{OM} and the biases vectors B^M , B^O . These system parameters are systematically calculated by the proposed method.

The polynomial function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ with m variables n degrees is represented by

$$f(\mathbf{x}) = c_{0\dots 0} + \sum_{1 \leq \nu_1 + \dots + \nu_m \leq n} c_{\nu_1 \nu_2 \dots \nu_m} x_1^{\nu_1} x_2^{\nu_2} \dots x_m^{\nu_m} = c_{0\dots 0} + C[\mathbf{x}^{\boldsymbol{\nu}}]. \quad (2)$$

Here, the m -dimensional multi-index $\boldsymbol{\nu} = (\nu_1, \dots, \nu_m)$ that satisfies the following conditions is used: $|\boldsymbol{\nu}| = \nu_1 + \dots + \nu_m$, $\boldsymbol{\nu}! = \nu_1! \dots \nu_m!$, and $\mathbf{x}^{\boldsymbol{\nu}} = x_1^{\nu_1} \dots x_m^{\nu_m}$.

¹ The superscript 'tr' of a matrix indicates the transpose of the matrix.

$C = [c_{\nu}; 1 \leq |\nu| \leq n]$ is the $1 \times N$ matrix of the polynomial coefficients² and $[\mathbf{x}^{\nu}] = [x_1^{\nu_1} \dots x_m^{\nu_m}; 1 \leq |\nu| \leq n]$ is the N dimensional vector of the variable terms³. N is the total of the possible case of $1 \leq |\nu| \leq n$, and is calculated by

$$N = \sum_{h=1}^n \binom{m+h-1}{h} = \sum_{h=1}^n \frac{(m+h-1)!}{(m-1)!h!}. \quad (3)$$

Let us derive a structure of TLFNNs approximating the polynomial function² on a compact set $K \subset \mathbb{R}^m$. For the constants $\alpha_i = [\alpha_{i1} \dots \alpha_{im}] \in \mathbb{R}^m$ and $\beta_i \in \mathbb{R}$, the Taylor expansion of $\sigma(\alpha_i \mathbf{x} + \beta_i)$ is expressed by

$$\sigma(\alpha_i \mathbf{x} + \beta_i) = \sigma(\beta_i) + \sum_{1 \leq |\nu| \leq n} \frac{\alpha_i^{\nu}}{\nu!} \frac{\partial^{|\nu|} \sigma}{\partial^{\nu_1} x_1 \dots \partial^{\nu_m} x_m}(\beta_i) \mathbf{x}^{\nu} + \sum_{|\nu|=n} o(\mathbf{x}^{\nu}), \quad (4)$$

where $o(\mathbf{x}^{\nu})$ is the infinitesimal for \mathbf{x}^{ν} . The notation $o(\mathbf{x}, n) = \sum_{|\nu|=n} o(\mathbf{x}^{\nu})$ is defined, then the following conditions are satisfied: for any $|\nu| \leq n$,

$$\lim_{\|\mathbf{x}\| \rightarrow 0} \frac{o(\mathbf{x}, n)}{\mathbf{x}^{\nu}} = 0, \quad (5)$$

⁴ and for any constants c_1 and c_2 ,

$$c_1 o(\mathbf{x}, n) + c_2 o(\mathbf{x}, n) = o(\mathbf{x}, n). \quad (6)$$

The partial derivatives of σ in (4) satisfy $\partial^{|\nu|} \sigma / (\partial^{\nu_1} x_1 \dots \partial^{\nu_m} x_m) = \sigma^{(|\nu|)}$, where $\sigma^{(|\nu|)}$ is the $|\nu|$ -th derivative of σ . Therefore, (4) is represented by

$$\sigma(\alpha_i \mathbf{x} + \beta_i) = \sigma(\beta_i) + \sum_{1 \leq |\nu| \leq n} \frac{\alpha_i^{\nu}}{\nu!} \sigma^{(|\nu|)}(\beta_i) \mathbf{x}^{\nu} + o(\mathbf{x}, n). \quad (7)$$

The N sets of α_i and β_i are chosen such that the following matrix P is non-singular. The next notations are also used. The $N \times m$ matrix and the N dimensional vector are defined by

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_N \end{bmatrix} = \begin{bmatrix} \alpha_{11} \dots \alpha_{1m} \\ \vdots \\ \alpha_{N1} \dots \alpha_{Nm} \end{bmatrix} \text{ and } \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}, \quad (8)$$

respectively. The $N \times N$ matrix of the Taylor expansion coefficients is defined by

$$P = \left[\frac{\alpha_i^{\nu}}{\nu!} \sigma^{(|\nu|)}(\beta_i); i = 1, \dots, N, 1 \leq |\nu| \leq n \right], \quad (9)$$

² For the multi-dimensional polynomial mapping $f : \mathbb{R}^m \rightarrow \mathbb{R}^q$, C is set to the $q \times N$ matrix. It is possible to discuss the method similarly as follows.

³ The order of the multi-index is appropriately set, and assumed that the order is kept in the following.

⁴ $\|\mathbf{x}\|$ indicates the norm of \mathbf{x} .

and the vector of the infinitesimals is defined by $\mathbf{o}(\mathbf{x}, n) = [o(\mathbf{x}, n), \dots, o(\mathbf{x}, n)]^{\text{tr}} \in \mathbb{R}^N$. Then, $\boldsymbol{\sigma}(\alpha\mathbf{x} + \boldsymbol{\beta}) = \boldsymbol{\sigma}(\boldsymbol{\beta}) + P[\mathbf{x}^{\boldsymbol{\nu}}] + \mathbf{o}(\mathbf{x}, n)$ is led from (7), where the activation mapping $\boldsymbol{\sigma} : \mathbb{R}^N \rightarrow \mathbb{R}^N$. When P is non-singular, there is the inverse matrix P^{-1} and the following condition holds:

$$[\mathbf{x}^{\boldsymbol{\nu}}] = P^{-1}\boldsymbol{\sigma}(\alpha\mathbf{x} + \boldsymbol{\beta}) - P^{-1}\boldsymbol{\sigma}(\boldsymbol{\beta}) + \mathbf{o}(\mathbf{x}, n). \quad (10)$$

The real parameter $\xi > 0$ that relates the approximation accuracy is prepared, and \mathbf{x} is replaced with $\xi\mathbf{x}$. Then, (10) is denoted by

$$[(\xi\mathbf{x})^{\boldsymbol{\nu}}] = P^{-1}\boldsymbol{\sigma}(\alpha\xi\mathbf{x} + \boldsymbol{\beta}) - P^{-1}\boldsymbol{\sigma}(\boldsymbol{\beta}) + \mathbf{o}(\xi\mathbf{x}, n). \quad (11)$$

In the left of the above equation, the condition $[(\xi\mathbf{x})^{\boldsymbol{\nu}}] = [[\xi^{|\boldsymbol{\nu}|}]][\mathbf{x}^{\boldsymbol{\nu}}]$ satisfies, where $[[\xi^{|\boldsymbol{\nu}|}]] = \text{diag}(\xi, \dots, \xi^{|\boldsymbol{\nu}|}, \dots, \xi^n)$ is the $N \times N$ diagonal matrix. There is the inverse matrix $[[\xi^{|\boldsymbol{\nu}|}]]^{-1} = \text{diag}(\xi^{-1}, \dots, \xi^{-|\boldsymbol{\nu}|}, \dots, \xi^{-n})$, thus

$$[\mathbf{x}^{\boldsymbol{\nu}}] = [[\xi^{|\boldsymbol{\nu}|}]]^{-1}(P^{-1}\boldsymbol{\sigma}(\alpha\xi\mathbf{x} + \boldsymbol{\beta}) - P^{-1}\boldsymbol{\sigma}(\boldsymbol{\beta}) + \mathbf{o}(\xi\mathbf{x}, n)) \quad (12)$$

holds from (11). Therefore, the next equation

$$\begin{aligned} f(\mathbf{x}) &= c_{0\dots 0} + C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}P^{-1}\boldsymbol{\sigma}(\alpha\xi\mathbf{x} + \boldsymbol{\beta}) \\ &\quad - C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}P^{-1}\boldsymbol{\sigma}(\boldsymbol{\beta}) + C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}\mathbf{o}(\xi\mathbf{x}, n) \end{aligned} \quad (13)$$

satisfies from (2) and (12).

On the other hand, the input-output mapping g of the TLFNN decided by the following coupling coefficients matrices and biases vectors is considered.

$$\begin{aligned} W^{MI} &= \xi\boldsymbol{\alpha} & W^{OM} &= C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}P^{-1} \\ B^M &= \boldsymbol{\beta} & B^O &= c_{0\dots 0} - C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}P^{-1}\boldsymbol{\sigma}(\boldsymbol{\beta}) \end{aligned} \quad (14)$$

Then,

$$f(\mathbf{x}) = g(\mathbf{x}) + C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}\mathbf{o}(\xi\mathbf{x}, n) \quad (15)$$

holds from (13). By the conditions (5) and (6), for any $\mathbf{x} \in K$,

$$\lim_{\xi \rightarrow 0} C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}\mathbf{o}(\xi\mathbf{x}, n) = 0 \quad (16)$$

satisfies. Then, for any $\varepsilon > 0$, there is $\xi > 0$ such that

$$|f(\mathbf{x}) - g(\mathbf{x})| = |C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}\mathbf{o}(\xi\mathbf{x}, n)| < \varepsilon \quad (17)$$

holds on the compact set K . Therefore, the TLFNN approximating the polynomial function is obtained. The number of the middle-layer neurons of the obtained TLFNN is $N^M = N$.

Moreover, $\partial f(\mathbf{x})/\partial x_i = \partial g(\mathbf{x})/\partial x_i + C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}\xi\mathbf{o}(\xi\mathbf{x}, n-1)$ for $i = 1, \dots, m$ and $\lim_{\xi \rightarrow 0} C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}\xi\mathbf{o}(\xi\mathbf{x}, n-1) = 0$ satisfy from (15) and (16). Similarly, for the multi-index $\boldsymbol{\mu} = (\mu_1, \dots, \mu_m)$, $1 \leq |\boldsymbol{\mu}| \leq n$,

$$\begin{aligned} \frac{\partial^{|\boldsymbol{\mu}|}}{\partial^{\mu_1}x_1 \dots \partial^{\mu_m}x_m} f(\mathbf{x}) &= \frac{\partial^{|\boldsymbol{\mu}|}}{\partial^{\mu_1}x_1 \dots \partial^{\mu_m}x_m} g(\mathbf{x}) \\ &\quad + C[[\xi^{|\boldsymbol{\nu}|}]]^{-1}\xi^{|\boldsymbol{\mu}|}\mathbf{o}(\xi\mathbf{x}, n-|\boldsymbol{\mu}|) \end{aligned} \quad (18)$$

and $\lim_{\xi \rightarrow 0} C[[\xi^{\boldsymbol{\nu}}]]^{-1} \xi^{|\boldsymbol{\mu}|} o(\xi \mathbf{x}, n - |\boldsymbol{\mu}|) = 0$ satisfy. Then, for any $\varepsilon > 0$, there is $\xi > 0$ such that (17) and

$$\left| \frac{\partial^{|\boldsymbol{\mu}|}}{\partial^{\mu_1} x_1 \dots \partial^{\mu_m} x_m} f(\mathbf{x}) - \frac{\partial^{|\boldsymbol{\mu}|}}{\partial^{\mu_1} x_1 \dots \partial^{\mu_m} x_m} g(\mathbf{x}) \right| = |C[[\xi^{\boldsymbol{\nu}}]]^{-1} \xi^{|\boldsymbol{\mu}|} o(\xi \mathbf{x}, n - |\boldsymbol{\mu}|)| < \varepsilon \quad (19)$$

hold on the compact set K . Therefore, it is shown that the obtained TLFNN can smoothly approximate the polynomial function.

3 Numerical Analysis

In this section, the example of the proposed method is explained⁵. In the following, the C^∞ function $\sigma(x) = \tanh x$ is used as the activation function. The similar results can be obtained for the activation functions $1/(1 + \exp(-x))$, \tan^{-1} and $\sin x$.

The polynomial function with two variables three degrees

$$f(x_1, x_2) = 2x_1 + 0.7x_1^2 - 0.3x_1x_2 - 0.6x_2^2 - 0.5x_1^3 + 0.2x_1^2x_2 + 0.4x_2^3 \quad (20)$$

on the compact set $K = \{(x_1, x_2); -5 \leq x_1 \leq 5, -5 \leq x_2 \leq 5\}$ is considered. Then, $m = 2$ and $n = 3$, thus the number of the middle-layer neurons is calculated

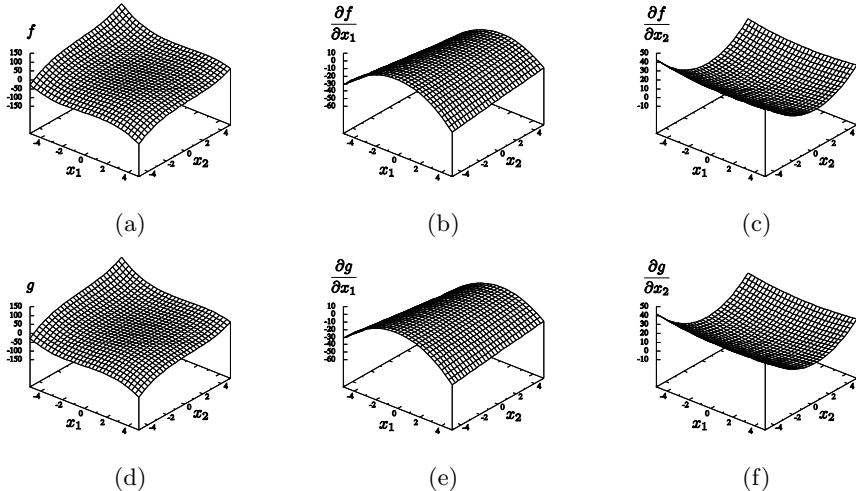


Fig. 1. Numerical results of the example. (a) Graph of f . (b) Graph of $\partial f / \partial x_1$. (c) Graph of $\partial f / \partial x_2$. (d) Graph of g . (e) Graph of $\partial g / \partial x_1$. (f) Graph of $\partial g / \partial x_2$.

⁵ To calculate the structure of the TLFNN, the Computer Algebra System 'Maxima' is used. In the following, the notation of the calculation values denotes to three decimal places though the accuracy of the calculation depends on Maxima.

by (3), $N = \sum_{h=1}^3 (2+h-1)!/((2-1)!h!) = 9$. The structure of the TLFNN is calculated by (14).

Figure 1 illustrates the numerical results of the example. The graphs of f , $\partial f/\partial x_1$ and $\partial f/\partial x_2$ are shown in Fig. 1(a), Fig. 1(b) and Fig. 1(c), respectively. When the parameter $\xi = 1/32$, the graphs of g , $\partial g/\partial x_1$ and $\partial g/\partial x_2$ are shown in Fig. 1(d), Fig. 1(e) and Fig. 1(f), respectively. The errors of these cases are less than 2.0 on the region K .

4 Conclusions

A simple calculation method of three-layer feedforward structures smoothly approximating polynomial functions is proposed. The structure of the three-layer feedforward network is decided by the coupling coefficients matrices W^{MI} , W^{OM} , and the biases vectors B^M , B^O . The proposed method shows that these system parameters can be calculated by (14). In this paper, though the structure of three-layer networks was treated, note that the structure of the feedforward networks with more than four layers is obtained from the cascade of the three-layer networks.

References

1. Bagheri, A., Karimi, T., Amanifard, N.: Tracking performance control of a cable communicated underwater vehicle using adaptive neural network controllers. *Applied Soft Computing* 10(3), 908–918 (2010)
2. Cybenko, G.: Approximation by superpositions of sigmoidal function. *Mathematics of Control, Signals and Systems* 2, 303–314 (1989)
3. Funahashi, K.: On the approximate realization of continuous mapping by neural networks. *Neural Networks* 2(3), 183–191 (1989)
4. Hornik, K., Stinchcombe, M., White, H.: Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks* 3, 551–560 (1990)
5. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks* 6, 861–867 (1993)
6. Meltser, M., Shoham, M., Manevitz, L.M.: Approximating functions by neural networks: a constructive solution in the uniform norm. *Neural Networks* 9(6), 965–978 (1996)
7. Scarselli, F., Tsoi, A.C.: Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results. *Neural Networks* 11(1), 15–37 (1998)
8. Souza, R.M.G.P., Moreira, J.M.L.: Neural network correlation for power peak factor estimation. *Annals of Nuclear Energy* 33(7), 594–608 (2006)
9. Suzuki, S.: Constructive function-approximation by three-layer artificial neural networks. *Neural Networks* 11, 1049–1058 (1998)
10. Toda, N., Funahashi, K., Usui, S.: Polynomial functions can be realized by finite size multilayer feedforward neural networks. In: 1991 IEEE International Joint Conference on Neural Networks, Singapore, vol. 1, pp. 343–348 (1991)

Neural Networks Training for Weapon Selection in First-Person Shooter Games

Stelios Petrakis and Anastasios Tefas

Department of Informatics, Aristotle University of Thessaloniki,
Box 451, 54124, Greece
tefas@aia.csd.auth.gr

Abstract. First person shooters is probably the most well known genre of the whole gaming industry. Bots in those games must think and act fast in order to be competitive and fun to play with. Key part of the action in a first person shooter is the choice of the right weapon according to the situation. In this paper, a weapon selection technique is introduced in order to produce competent agents in the first person shooter game Unreal Tournament 2004 utilizing the Pogamut 2 Game-Bots library. The use of feedforward neural networks is proposed, trained with back-propagation for weapon selection, showing that that there is a significant increase at the performance of a bot.

Keywords: Computational intelligence and Games, Artificial Intelligence and games, First person shooter games, bots, computational intelligence, neural networks.

1 Introduction

The majority of computer games nowadays, contain some basic AI functionality which makes use of scripted behaviors and finite state machines (FSMs), suffering by this way from a lot of basic gameplay issues, as human players can predict NPC behaviors and easily exploit them.

Another tough challenge for game AI is to make a bot feel like human, by representing more “human” behaviors, which can often be random and imperfect. Computational intelligence is trying to fill the gap for that need with mimics, memetics, neural networks, fuzzy logic and evolutionary algorithms which can be adjusted and make bots look “human”.

In first person shooters, more like any other genres, the fast paced and face to face gaming style, require even more human behavior, especially when the opponents are designed to look like humans, thus fooling the human player to expect them to act with human intelligence.

2 Problem

For any given game moment, the main problem of training of a game agent in a first person shooter is the ability to respond to the sensory data, with ways that make him act like a human player.

This problem can be easily modeled with a back propagation neural network, as neural networks are systems that can be trained in order to learn a certain behavior.

In the proposed method of this paper, the sensory data can be used as input vector for those neural networks, which practically involves everything an agent can see, hear or sense in any other way. Input vectors are normalized in $(0, 1)$ space, in order to form a uniform data set and are picked carefully judging from their relevance to the problem each neural network is trying to solve.

The output vector for those neural networks is the action vector, as it will eventually dictate how the bot should act for a given input. Usually action vectors consist of a single value in $(0, 1)$ space.

This paper focuses in the weapon selection decision a bot has to make several times during a battle in a first person shooter game. For this technique, the bot was trained separately for each weapon, having three input values, the angle and distance of the two players (bot and enemy) as well as the velocity of the enemy. The output value of those neural networks (one for each weapon), would calculate the estimated damage that the weapon might do. So, each time, the agent would pick the weapon that would make the maximum damage and fire at the enemy.

3 Related Work

Using Unreal Tournament 2004 and Gamebots API as a testbed, Ronan Le Hy, et al. [2] were able to apply Bayesian programming to behaviors of bots. They applied probabilistic behavior recognition using a Bayesian model, showing that this method can lead to condensed and easier formalization of finite state machine-like behavior selection, and lend itself to learning by imitation, in a fully transparent way for the player.

Spronck, et al. [5], used dynamic scripting to enable Non-Player Characters (NPCs) to evolve according to different players. NPCs created in different time had different rules and could improve the satisfactory level of human players as their opponents demonstrate a dynamic behavior.

Cook, et al. [6] used graph-based relational learning algorithm to extract patterns from human player graphs and applied those patterns to agents, showing by this way how human knowledge could be transferred.

Kim [11] defined a finite state transition machine to switch behaviors of the agent based on context sensitive stimulations received from the game.

Di Wang, et al. [4], participated in the Bot Prize 2008, using Pogamut library to create bots, implementing FALCON technique, a self-organizing neural network that performed reinforcement learning. Bots were able to learn in real-time without any human intervention.

Hirono et al. [3] developed a bot in Gamebots and Unreal Tournament, winning the second place in Bot Prize 2008, behaving based on Finite State Automaton having two states: the item-collection state (the initial state) and the battle state. Their bot was able to fool some of the judges into believing it was a real person, using heuristic methods.

Unlike the works described above, the technique introduced in this paper uses neural networks to train bots offline and then applies the trained neural networks in realtime situations in order to measure the improvement on their performance.

4 Platform

4.1 Unreal Tournament 2004

Unreal Tournament 2004 is primarily a multiplayer game. Gamers can train themselves against bots with multiple difficulty levels, in order to be competitive against human opponents. Those bots are using Unrealscript with predefined scripted events in order to achieve victory. They are neither learning from their mistakes nor trying to improve their overall performance, contrary to the bot design that is being presented in this paper. There are eight levels of bot skills, each one increasingly more difficult than the previous: novice, average, experienced, skilled, adept, masterful, inhuman and godlike.

This paper focuses in only a single game type, Deathmatch, to train and test the bots, due to its straight forward and single objective nature. Unreal Tournament 2004 offers players with a wide range of weapons, that demonstrate different abilities. For the purposes of this paper, nine from the total 17 weapons of the game has been chosen, as those weapons can be found in nearly every single multiplayer level of the game.

4.2 Pogamut 2 Gamebots

For the experiments explained in this paper, the second version of Pogamut, Pogamut 2 has been used. Pogamut 2 has four components: Gamebots, Server, Client, and IDE. Gamebots is managing the information from Unreal Tournament 2004 and the execution of commands. Server is running the deathmatch type of game in Unreal Tournament 2004 and for testing purposes is running in the same machine as the client. Client is handling the connection to the server, parsing messages and providing libraries of atomic actions and basic sensors. The Integrated Development Environment (IDE) used for Pogamut 2 is Netbeans along with the appropriate plug-in which contains features such as server controller and log viewers.

5 Weapon Selection

The approach described in this paper uses nine different neural networks and the back propagation method for training.

In order to decide which weapon to pick, agent must be able to predict the approximate damage every weapon could do based on the situation. For this reason nine neural networks NN^n have been created, one for each weapon n with $n = 1 \dots 9$. Each neural network NN^n has an input layer \mathbf{x}^n of 3 neurons, where $\mathbf{x}^n = (x_1^n, x_2^n, x_3^n)$, with $x_i^n \in [0, 1]$ and x_1^n representing the distance

between the two players, x_2^n the angle between them and x_3^n the velocity of the enemy, one hidden layer with 50 neurons and an output layer with one neuron y^n representing the estimated damage of the weapon n .

5.1 Neural Network Initialization

Hecht-Nielsen [7] proved that one hidden layer is sufficient to approximate a bound continuous function to an arbitrary accuracy. However, one hidden layer may result in an excessive number of neurons used in the hidden layer. This is the reason, every neural network presented in this paper, has been modeled using 50 neurons in its single hidden layer.

As an activation function for the neural network the hyperbolic tangent $\varphi(\nu) = \tanh\left(\frac{\nu}{2}\right) = \frac{1-\exp(-\nu)}{1+\exp(-\nu)}$ has been used and the learning rate η of the neural network has been set to 0.5.

5.2 Training Phase

The bots of this paper have been trained by implementing a scripted behavior which dictated that they had to shoot the enemy every time he was visible. For each encounter, the bot was facing one Unreal Tournament pre-made scripted bot which was configured for two different tests, in the first (novice) and the seventh (inhuman) level of difficulty and will be known as '*enemy*' from now on.

Training phase collected approximately 30.000 input states for each weapon, which were used to initially train 9 different, randomly initialized, neural networks. Every time the bot was damaging the enemy, the following variables were being used: the resulting damage normalized in $[0, 1]$ space as the desired output z of the neural network and the variables: distance x_1 , velocity x_2 and angle x_3 of that particular moment, as input variables.

Neural networks have been trained using the back propagation method, by passing the damaged caused as desired output and the three above values as inputs, all normalized at $[0, 1]$ space.

5.3 Offline Training Phase

After the data collection process was complete for every weapon, two files were created for each weapon: One with all the input and output values of the training set and one with the weights of the first generation neural networks.

Having nine pairs of files, offline training has been applied to further train the neural networks for 1.000 epochs. After the offline learning process, the neural networks were ready for testing.

5.4 Testing Phase

During the testing phase, the nine trained neural networks were initiated with the structure exported from the offline training and were integrated into the bot

decision making process. The bot had a pretty simple instruction set: In idle mode it had to run around the map collecting any health, weapon and armor power up but the moment it come across an enemy, it calculated which weapon of his inventory could do the maximum damage and fire to him.

Let N_l^k+ be the number of frags of a bot trained versus an enemy of level k , when it faces an enemy level l in the simulated environment.

The performance curve of such a bot, based on the ratio of the number of frag of this bot to the number of frags of the enemy N_l^k- , can be presented like this:

$$P_l^k = \frac{N_l^k+}{N_l^k-}$$

In this way, when $P_l^k > 1$, bot performs better than the enemy with level l .

Tests measured the performance of the bot against Unreal Tournament enemies with different level of difficulty each time. As noted previously, the bot was not only trained with Unreal Tournament enemies of $k = 1$ (easy skill), but also with $k = 7$ (godlike skill).

Test results showed a major performance improvement. More specifically, when the bot was trained with an enemy of level $k = 7$, it could perform better against enemies of maximum level $l = 2$, whereas when it was trained with an enemy of level $k = 1$ it could only perform well against enemies of same level. It must be noted that just by applying the weapon selection method in a bot which does nothing else other than randomly moving through the map, an immediate major improvement of the bot score was observed, compared with a random weapon selection bot.

In the graph at Figure 1, the dotted line represents the frag ratios P_l^R of a random bot (no neural network attached) versus different levels of the enemy ($l = 1 \dots 7$), the dashed line represents the frag ratios P_l^1 of the neural network bot presented in this paper trained with an enemy of level $k = 1$, whereas the

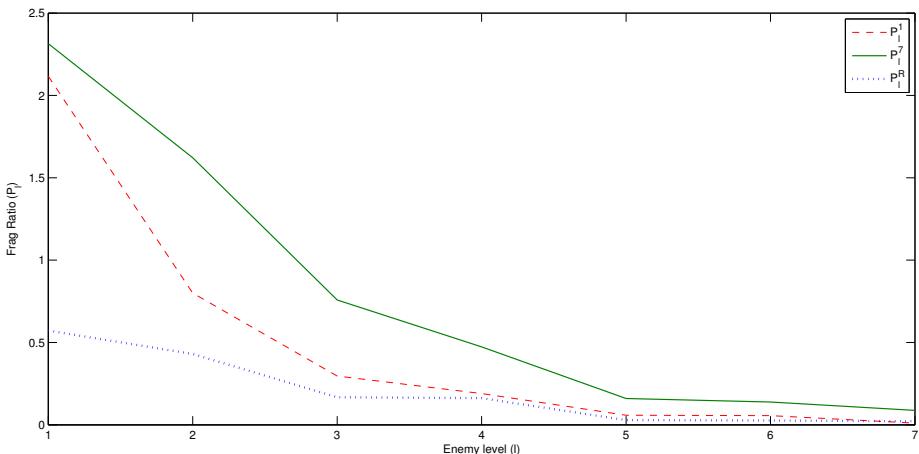


Fig. 1. Frag ratio P_l for a random bot P_l^R / a bot trained with an enemy level $k = 1$ P_l^1 / a bot trained with an enemy level $k = 7$ P_l^7 for $l = 1 \dots 7$

solid line represents the frag ratios P_l^7 of the neural network bot trained with an enemy of level $k = 7$.

When the bot is trained with a much more difficult enemy ($k = 7$), it is not only possible to win an enemy with level $l = 2$, but it can also improve its score with the level $l = 1$ enemies.

So, in conclusion a bot trained to face more difficult enemies, can perform better in battle from a bot that has been trained to face easier enemies. So the measure of performance of such a bot versus enemies of level l is the frag ratio $P_l^k = \frac{N_l^k +}{N_l^k -}$.

6 Conclusions

This paper proposed a computational intelligence method using neural networks, in order to develop a bot which can be trained while playing Unreal Tournament 2004 using Pogamut 2 Gamebots library, trying by this way to achieve higher performance than the average dummy bot. The introduced method involved a weapon selection module, where the bot was trained to choose the weapon to fire depending on the condition it was in. It was shown that by training the bot for the weapon selection module using single-hidden layer feedforward neural networks, led to a significant increase in the performance. There was also an increase of bot's performance while it was facing even more difficult enemies.

References

1. Kim, I.-C.: UTBot: A Virtual Agent Platform for Teaching Agent System Design. *Journal of Multimedia*, 48–53 (2007)
2. Le Hy, R., Arrigoni, A., Bessiere, P., Lebeltel, O.: Teaching Bayesian behaviours to video game characters. *Robotics and Autonomous Systems* 47, 177–185 (2004)
3. Hirono, D., Thawonmas, R.: Implementation of a Human-Like Bot in a First Person Shooter: Second Place Bot at BotPrize 2008. In: Proc. Asia Simulation Conference 2009, JSST 2009 (2009)
4. Wang, D., Subagdja, B., Tan, A., Ng, G.: Creating Human-like Autonomous Players in Real-time First Person Shooter Computer Games. In: Proc. IAAAI 2009 (2009)
5. Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., Postma, E.: Adaptive Game AI with Dynamic Scripting. In: Machine Learning, pp. 217–248 (2006)
6. Cook, D.J., Holder, L.B., Youngblood, C.M.: Graph-Based Analysis of Human Transfer Learning Using a Game Testbed. *IEEE Transaction on Knowledge and Data Engineering* 19, 1465–1478 (2007)
7. Hecht-Nielsen, R.: Theory of backpropagation neural network. In: Proc. of the Int. Conf. of Neural Networks, vol. I, pp. 593–611 (1989)

Efficient Confidence Bounds for RBF Networks for Sparse and High Dimensional Data

Abner Rodrigues Neto¹, Mauro Roisenberg¹, and Guenther Schwedersky Neto²

¹ Universidade Federal de Santa Catarina

Florianópolis, Brasil

² Petróleo Brasileiro S.A.

Rio de Janeiro, Brasil

Abstract. Artificial Neural Networks have been used for function approximation and pattern recognition in a variety of domains. However, due to its empirical nature, it is difficult to derive an estimate of neural network's accuracy. There are in the literature a number of proposed methods to calculate a measure of confidence to the output of neural networks but in general these approaches require some strong assumptions which are rarely observed in real problems. This paper analyzes and extends the Validity Index Network, a model derived from radial basis function network that calculates the confidence of its outputs, we remove its restrictions in density calculations, specially in high dimensional input spaces, and improve the probability coverage of the prediction levels when the training data have variable density.

Keywords: neural networks; density estimation; prediction interval; RBF.

1 Introduction

Due to its empirical nature, is difficult to understand when an artificial neural network (ANN) is extrapolating or estimating the output to a region whose training data were insufficient to achieve a good approximation. Measures of overall performance that are commonly used to evaluate the performance of neural network, as the mean square error, are not able to recognize areas where the network's response can be contaminated by uncertainties due to factors such as model error due noisy or sparse training data. In order to deal with this problem, the proposed solution is to calculate a way to measure reliability of the model, for example, the prediction intervals (PI).

In general, the methodology for PI calculation is specific to the neural network architecture. There are basically two broad approaches: the local approximation based approach where PI estimation is done through linear regression and, the global approximation based approach that uses nonlinear regression to calculate the PI.

Learning in some ANN architectures such as radial basis function networks (RBF) are based in a local approximation approach. Validity Index Network

(VINet), is an extension of the RBF network proposed by Leonard et al. [1] that calculates the PI to its output, and others confidence measures.

Moreover, global approximation based networks, such as Multilayer Perceptrons (MLP), do not have this concept of local neighborhood and therefore can not be easily extended to incorporate the calculation of PI. Some solutions to calculate the PI in MLP networks have been proposed in the literature [2], [3], [4], [5]. However, these algorithms assume strong restrictions that must be satisfied, such as: the number of training points should tend to infinity [5], residuals must be independent and distributed according to a Gaussian with zero mean [2] and, the network must be trained until convergence [4].

These conditions are not always true in real problems and not need to be satisfied in VINet, which is much simpler. Moreover, in comparative studies it was observed that the size of the prediction interval calculated in VINet does not always match the distribution of training data or fail to reach a desired coverage probability [3], [6]. Furthermore, the way the density is calculated on VINet can lead to unsatisfactory results, especially when using high-dimensional input data [7].

The objective of this work is to analyze the VINet in order to propose, develop and test some techniques to overcome this limitations. We purpose to change the way that data density is calculated by VINet using self-organizing networks, an approach that is independent of the dimensionality of the data. Next we incorporate this density value in the calculation of PI, in order to correct the shortcomings of the VINet, to obtain a PI compatible with the expected coverage probability.

2 Validity Index Network

The VINet is an RBF network extension that calculates the PI in the output [1]. The calculation of PI, for an input x , is:

$$PI(x) = \frac{\sum_{j=1}^m v_j(x) PI_j}{\sum_{j=1}^m v_j(x)} \quad (1)$$

where m is the total number of neurons in the hidden layer, $v_j(x)$ is the activation of the j -th neuron from hidden layer and PI_j is the local PI calculated by:

$$PI_j = t_{n_j-1}^{\alpha/2} S_j \left(1 + \frac{1}{n_j} \right)^{1/2} \quad (2)$$

where $t_{n_j-1}^{\alpha/2}$ is the inverse of the Student t cumulative distribution function with $n_j - 1$ degrees of freedom, evaluated at $\alpha/2$ and n_j is the number of training points associated with each hidden neuron j , given by:

$$n_j = \sum_{i=1}^Q v_j(x_i) \quad (3)$$

where Q is the number of training points. S_j is the standard deviation for each neuron, calculated by:

$$S_j^2 = \frac{\sum_{i=1}^Q v_j(x_i)(y_i - f(x_i; \hat{\theta}))^2}{n_j - 1} \quad (4)$$

where $(y_i - f(x_i; \hat{\theta}))^2$ is the difference between desired value y_i and the value calculated by the network $f(x_i; \hat{\theta})$, for the training data x_i . In VINet, two more confidence outputs are calculated: the density and the extrapolation flag. The probability density for the given input x is calculated by using Parzen windows [8]. The density for a point x is given by:

$$p(x) = \frac{\sum_{j=1}^m v_j(x)p_j}{\sum_{j=1}^m v_j(x)} \quad (5)$$

where p_j is calculated during training:

$$p_j = \frac{\sum_{i=1}^Q v_j(x_i)}{n(\pi^{1/2}r)^N} \quad (6)$$

where N is the dimension of the data and i is the index of each training data.

The disadvantage in using the Parzen window is that when calculated for high dimensional data, the resulting density value will approach zero and cease to be meaningful [7]. Another problem is that we can increase the reliability of density estimation through Parzen window method as we increase the number of neurons in the hidden layer, but there is no guarantee that the optimal choice of the number of neurons to density estimation is equal to the best number of hidden neurons in what concerns good generalization capacity of the network [9] [7].

In order to overcome restrictions presented by the Parzen window method, in the next section we present an alternative approach that is able to estimate density independently of the input data dimension.

3 Proposed Methodology for Density Estimation

The main idea of the proposed methodology is based in the fact that the distribution of weights in self-organizing map (SOM) tends to the distribution of training data. Regions with more data tends to be represented by more neurons that are activated more frequently [10]. Thus, we can use the SOM to calculate the probability distribution function of the data. One way to do this is using the concept of *mixture models*, which represents the distribution by a linear combination of some kernel functions [11]. The density for an input x can be calculated as:

$$p(x) = \sum_{j=1}^m P(j)p(x|j) \quad (7)$$

where m is the number of neurons of the SOM, $P(j)$ are the weights of the mixture and can be interpreted as the a priori probability that a data point belongs to one Voronoi diagram cell corresponding to neuron j and $p(x|j)$ represents the conditional probability of the desired vector x to the j -th kernel. The probabilities $P(j)$ must satisfy the conditions:

$$\sum_{j=1}^m P(j) = 1 \quad (8)$$

$$0 \leq P(j) \leq 1 \quad (9)$$

Similarly the probabilities $p(x|j)$ must be standardized to ensure that:

$$\int p(x|j)dx = 1 \quad (10)$$

For the kernel functions, one common choice is the Gaussian, where the center is the weight of the neuron and the radius can be calculated using the Euclidean distance between the weight of a neuron j to the k -th nearest neurons weights. A suggestion given by Shao et al. [3] is incorporate the density to the PI calculation so its properly reflects data density and gives an appropriate confidence level. The PI may be wider or narrower depending upon the density and can be given as:

$$PI_f(x) = \frac{2PI(x)}{1 + \frac{\hat{p}(x)}{p_{max}}} \quad (11)$$

where \hat{p} is the density estimated and p_{max} is the maximum value of p .

4 Case Studies

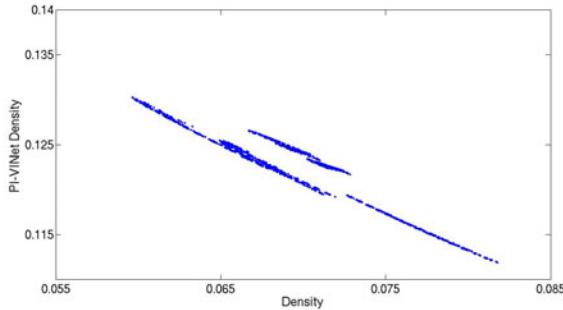
In order to compare the accuracy of PI calculated by VINet with the PI calculated by the our proposed methodology, we present two tests. This comparison was done by training the network and estimating the PI for a 95% confidence level, then test data were presented to the network and the Prediction Interval Covarage Probability (PICP) was calculated, ie, the percentage of test data actually located in the region bounded by the PI. Altogether 100 VINet were instantiated in each test and trained with the same data set. Then the PICP was calculated for each one and finally averaged for all networks.

The first test was a simple case study proposed by [1]. The function used to generate the data was given by the following equation: $y(x) = 0.5 \sin(1.5\pi x + \pi/2) + 2.0 + v$, where v was a Gaussian noise with zero mean and standard deviation 0.1. The training set was formed by 31 data points within the range $[-1, 1]$ chosen accordingly to a non uniform probability function.

Table 1 shows the real PICP obtained by each approach to 1000 test points submitted to the networks. In the case of VINet, it's coverage probability was below the desired coverage of 95%. The PI calculated using Leonard et al. [1]

Table 1. PICP for each method

	VINet	VINet-density
First Experiment	84.816	95.192
Second Experiment	76.87	91.33

**Fig. 1.** PI calculated with the proposed methodology versus density

proposed methodology hardly reflects the density of data and maintains an almost constant PI either in regions with more training data as in those with fewer ones. The PI with density incorporated in the calculation was able to reflect the distribution of training data and obtained a performance very close to the desired in PICP.

The second case study was used to test the proposed methodology in a high-dimensional input space, where density calculation using traditional Parzen windows would lead to unsatisfactory results. This is a real world problem was taken from Proben1 database [12], as a multi-input single-output problem which intend to estimate the energy consumption of a building based on 14 input variables. The data set was composed by 4208 data patterns, 50% of which were used for training and 50% for tests. The network had 40 neurons in the hidden layer and the SOM was created with 10 by 10 neurons in the competitive layer.

As shown in Table 1 the real PICP obtained using the VINet was below the desired confidence level. Using the proposed methodology were density were calculated with a SOM and then incorporated in PI calculation, the confidence bounds became wider and gave a more realistic measure of the real coverage, thus reflecting the uncertainty of network prediction. Figure 1 plots the density versus PI calculated with the proposed method for the test points. The behavior of the PI, as expected, presents a narrow PI in high density regions and a wider one in low density regions. There can be seen in the region with density around 0.070, two distinct PI. These two distinct PI in the same density region can be explained observing the error variance of the training set: the points with greater PI are from a region with greater noise.

5 Conclusions

This paper presents some options to correct VINet deficiencies. We proposed a way to calculate the density that is independent of data dimension, thus solving the density value tending to zero for high-dimensional data. Furthermore, it was shown that in some cases the PI calculated by VINet does not match the desired level of confidence, and also does not reflect the density of training data. With the incorporation of density in the PI, the points that fall in regions where the training was not effective will have a wider PI, indicating less confidence in the forecasts for these regions. These changes are welcome in practical and industrial applications, where we need to make sure that the network will continue to generate reliable outputs after training. When the network receive data much different than the ones presented in the training, the PI will be wider, indicating a novelty for the network and therefore it should be retrained with more data from this region. With these features, users have more control of the network for training and more confidence in approximations.

References

1. Leonard, J.A., Kramer, M.A., Ungar, L.H.: A neural network architecture that compute its own reliability. In: Computers Chem. Engng. (1992)
2. Chryssolouris, G., Lee, M., Ramsey, A.: Confidence interval prediction for neural network models. IEEE Trans. Neural Networks 7, 229–232 (1996)
3. Shao, R., Martin, E.B., Zhang, J., Morris, A.J.: Confidence bounds for neural network representations. Computers Chem. Engng. 21, 1173–1178 (1997)
4. Veaux, R.D., Schweinsberg, J., Shellington, D.: Prediction intervals for neural networks via nonlinear regression. Computers Chem. Engng. 21, 1173–1178 (1998)
5. Hwang, J.T.G., Ding, A.A.: Prediction intervals for artificial neural networks. J. American Statistical Association 92(438), 748–757 (1997)
6. Yang, L., Kavli, T., Carlin, M., Clausen, S., de Groot, P.: An evaluation of confidence bound estimation methods for neural networks. In: ESIT 2000 (1991)
7. Wedding, D.K., Cios, K.J.: Certainty factors versus pazen windows as reliability measures in rbf networks. Neurocomputing 19, 151–165 (1997)
8. Parzen, E.: On estimation of a probability density function and mode ann. Math. Statist. 33, 1065–1076 (1962)
9. Bishop, C.M.: Novelty detection and neural network validation. In: IEE Proc.-Vis. Image Signal Process., vol. 141 (1994)
10. Holmstrom, L., Hamalainen, A.: The self-organizing reduced kernel density estimator. In: IEEE International Conference on Neural Networks, pp. 417–421 (1993)
11. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford Press, Oxford (1995)
12. Prechelt, L., Informatik, F.F.: Proben1 - a set of neural network benchmark problems and benchmarking rules. Technical report (1994)

Large Scale Problem Solving with Neural Networks: The Netflix Prize Case

Nicholas Ampazis*

Department of Financial and Management Engineering,

University of the Aegean, 82100 Chios, Greece

n.ampazis@fme.aegean.gr

Abstract. The significant advances in artificial neural network research during the last two decades has rendered neural network models well established data analysis techniques. In this paper we examine the performance of a number of neural network models when applied to the Netflix Prize dataset, which is a large scale (more than 100 million training patterns) high-dimensional, and very sparse dataset. Our study comprises of the combination of different neural network approaches that are presented in the main part of the paper.

1 Introduction

In October 2006 Netflix, one of the largest online DVD rental companies in the USA, released a large movie ratings dataset and challenged the data mining, machine learning and computer science communities to develop systems that could beat the accuracy of their in-house developed recommendation system (Cinematch) by 10% [1]. Netflix provided 100,480,507 ratings (on a scale from 1 to 5 integral stars) along with their dates from 480,189 randomly-chosen, anonymous subscribers on 17,770 movie titles, thus making it by far the largest ratings dataset becoming available to the recommender systems research community. In order to render the challenge more interesting, the company announced that it would award a Grand Prize of \$1M to the first team able to attain this goal. In July 2009, after almost three years and more than 43,000 entries from over 5,100 teams in over 185 countries [2], the Netflix Prize Contest goal was surpassed by two teams, "BellKor's Pragmatic Chaos" and "The Ensemble", which actually tied on the accuracy improvement goal (10.06%). In this situation, under the contest rules, the tie breaker became the order of the submissions and "Bellkor's Pragmatic Chaos" were announced winners, as they submitted their results just 20 minutes earlier than "The Ensemble" before the end of the contest [3].

During our research efforts with the Netflix dataset it soon became clear that most of the latent factor, clustering, and nearest neighbors approaches to the Netflix prize [4,5,6] can be implemented with neural network models that are presented in this paper, which can provide good accuracy levels and are computationally efficient.

* Also member of "The Ensemble".

2 Matrix Factorization

The idea behind matrix factorization techniques is very simple. We define the corresponding matrix factorization $\hat{\mathbf{X}}_k$ of the user-by-item matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ as $\hat{\mathbf{X}}_k = \mathbf{C}\mathbf{M}$ ($\mathbf{C} \in \mathbb{R}^{I \times K}$ and $\mathbf{M} \in \mathbb{R}^{K \times J}$). For the recommendation problem, \mathbf{X} has many unknown elements which cannot be treated as zero. For this case, the approximation task can be defined as follows:

$$e_{ij} = x_{ij} - \hat{x}_{ij}, \quad \hat{x}_{ij} = \sum_{k=1}^K c_{ik} m_{kj} \quad (1)$$

where \hat{x}_{ij} denotes how the i th user would rate the j th movie, according to the model, and e_{ij} denotes the error on the (i, j) th example (training pattern). In order to minimize this error (and consequently the error over all training patterns) we can apply the *stochastic* version of the gradient descent method on e_{ij}^2 to find a local minimum. Hence the elements of \mathbf{C} and \mathbf{M} can be updated as follows:

$$c'_{ik} = c_{ik} + \eta \cdot 2e_{ij} \cdot m_{kj}, \quad m'_{kj} = m_{kj} + \eta \cdot 2e_{ij} \cdot c_{ik} \quad (2)$$

where η is the learning rate. To better generalize on unseen examples, we can apply regularization with factor λ in order to prevent large weights:

$$c'_{ik} = c_{ik} + \eta \cdot (2e_{ij} \cdot m_{kj} - \lambda \cdot c_{ik}), \quad m'_{kj} = m_{kj} + \eta \cdot (2e_{ij} \cdot c_{ik} - \lambda \cdot m_{kj}) \quad (3)$$

2.1 Matrix Factorization as Feed-Forward Neural Network Training

The matrix factorization model can be considered as learning within a multi-layer feed-forward network framework. The network has I inputs, K hidden nodes and J outputs, and all neurons have identity activation functions. Customer features c_{ik} can be considered as the weights between the i th input and the k th hidden neuron, and movie features m_{kj} as the weights between the k th hidden neuron and the j th output neuron. For the (i, j) th rating we can set the input x_i to 1 and all other $x_{k \neq i} = 0$. In this case the network can be trained with the *online* backpropagation algorithm (with regularization) using equations (3). In the testing phase, we set the input as in the training phase, and y_j ($j = 1 \dots J$) predicts the active user's rating on the j th movie. We refer to this method as Neural Network Matrix Factorization, or shortly, as NNMF. The network can be modified with the addition of biases (representing baseline user and movie ratings) [4]. Another modification is to include non-linear activation functions (e.g. sigmoid) in the hidden layer or in both the hidden and the output layer. In this case we refer to the method as NNMF-S and NNMF-SS respectively.

3 Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) were introduced in the context of recommender systems in [7]. In the input layer of an RBM we present a $\mathbf{V} \in \mathbb{R}^{K \times m}$

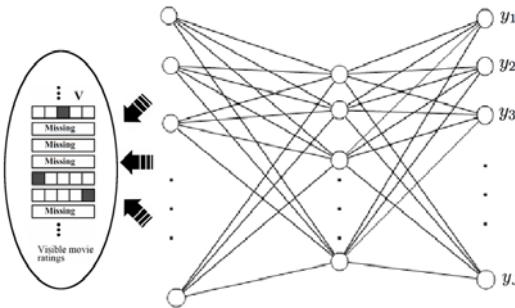


Fig. 1. Auto-encoder Neural Network for Recommendations

binary matrix, where m is the number of movies and K is the number of integer rating values. We set $v_i^k = 1$ if the user rated movie i as k and 0 otherwise. Given the hidden unit activations h_j , the values of each column of the observed visible rating matrix \mathbf{V} can be predicted by a conditional multinomial distribution (softmax) as:

$$p(v_i^k = 1 | \mathbf{h}) = \frac{\exp(b_i^k + \sum_{j=1}^F h_j W_{ij}^k)}{\sum_{l=1}^K \exp(b_i^l + \sum_{j=1}^F h_j W_{ij}^l)} \quad (4)$$

where W_{ij}^k is a symmetric weight between hidden feature j and rating k of movie i , and b_i^k is the bias of rating k for visible movie movie i and . Learning in this model is achieved by an approximation to the gradient of an objective function called "Contrastive Divergence" (CD) as suggested in [8]. A significant performance boost can be further achieved in RBMs by making the hidden units conditional on which movies the current user has rated but for which we may not know the exact rating that the user has given. The training of these *conditional* RBMs is accomplished by simple modifications of the CD rule. Further details can be found in [7]. We refer to this method as COND-RBM.

4 Auto-encoder Neural Network

The symmetric nature of RBM's weights W_{ij}^k between hidden feature j and rating k of movie i , essentially renders them auto-encoder networks. It would therefore be interesting to modify the learning model of section 2.1 to the one shown on Figure 1. As with RBMs, we can represent each user by the movies that he has rated. The network has $K \times J$ inputs, H hidden nodes, but also utilizes an output layer with J neurons. For the (i, j) th rating, in the input layer we present the binary matrix $\mathbf{V} \in \mathbb{R}^{K \times J}$ corresponding to user i , and calculate the activation y_j which predicts the active user's rating on the j th movie. The network can be trained with *online* backpropagation (with regularization) using equations (3). Similar to our abbreviations introduced in section 2.1, we refer to this method, with non-linear activation functions, as NN-AUTO-SS.

5 User Clustering with Self Organizing Maps

User clustering can be implemented with Self Organizing Maps (SOMs). As with RBMs, users can be represented by the movies that they have rated. However due to the sparsity in user ratings and the nature of the SOM training rule, a naive implementation with J inputs (where J is the total number of movies) is not going to work as only a fraction of the SOM codebook vectors's components will be affected by the update rule. An alternative is to represent users by the $V \in \mathbb{R}^{K \times J}$ matrix of a trained RBM with v_i^k given by the softmax activations of equation (4) to get the probability distribution over the K ratings for each movie i rated by the user in question. A schematic diagram of such a network, which utilizes a 2-dimensional SOM lattice is shown in Figure 2. Once the SOM has been trained an active user's u rating for an unseen movie m can be evaluated as follows:

- Find the k clusters that are most similar to user u . Calculate the membership of the user to each cluster according to a suitable similarity metric.
- The predicted rating P_{vm} for user u on movie m is the weighted average of the k clusters' ratings, given by $P_{um} = \frac{\sum_k r_{mk} w_{uk}}{\sum_k w_{uk}}$, where r_{mk} is the average rating of movie m in user cluster k and w_{uk} is the membership (weight) of user u in cluster k .

Subsequently we normalize the predictions over the K possible ratings. We refer to this method as RBM-SOM.

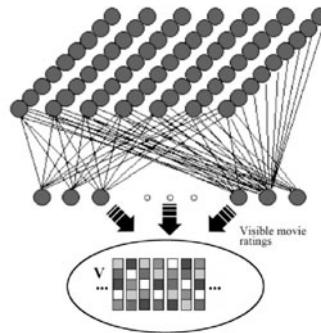


Fig. 2. User Clustering SOM Model

6 Radial Basis Function Networks

Users's ratings for unseen movies may also be predicted using trained Radial Basis Function (RBF) Networks. Training in RBFs may be achieved by utilizing gradient descent for weights between hidden and output layer and a separate clustering scheme for learning the centers c_j of the hidden nodes. The learning of the RBF hidden neuron centers was accomplished by utilizing a *trained* SOM as described

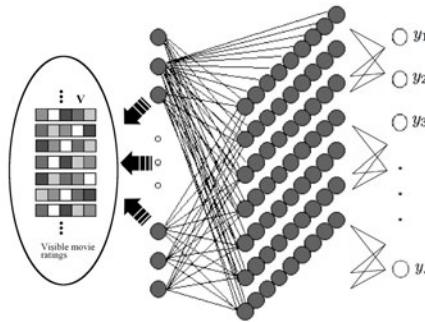


Fig. 3. SOM trained RBF

in section 5, that is we represented users by the $\mathbf{V} \in \mathbb{R}^{K \times J}$ matrix of a trained RBM with v_i^k given by the softmax activations of equation (4). For the RBF training we utilized a 2-dimensional SOM lattice and J outputs as depicted in Figure 3. The SOM nodes were effectively considered as the hidden nodes of the RBF network. The variances for all hidden nodes were set to $\sigma_j = \frac{D}{2P}$ ($\forall j$), where D is the maximum distance among the centers \mathbf{c}_j , and P the number of training patterns. The hidden to output weights w_{kj} were trained by *online* backpropagation with regularization. We refer to this method as RBM-SOM-RBF.

7 Experimental Results

As part of the training data, Netflix also provided validation data, a subset of the training set (known as "Probe" set), containing 1,408,395 known ratings, which was obviously excluded during training. In addition to the training and validation data, Netflix also provided a "Qualifying" set containing 2,817,131 user/movie pairs with the ratings withheld. As a performance measure Netflix had selected the Root Means Square Error (RMSE) criterion between the actual and predicted scores. To reduce unintentional fine-tuning on the qualifying set performance was assessed by submitting predicted ratings to Netflix which then reported the RMSE on an unknown half of the qualifying set ("Quiz" set). As a baseline, Netflix provided the RMSE score of its own system (Cinematch) on the quiz data, which was 0.9514 [1]. Table II reports the best RMSE score attained by the various NNMF, RBM, SOM based clustering, RBF methods, and their combinations, on both the probe and the quiz sets. In parenthesis we report the number of hidden nodes (features) used in each particular implementation. Note that in all methods we have included bias terms. In the same table we also report the training time of each approach on the same machine (a 2.5GHz quad core Pentium CPU with 4G RAM running the Ubuntu 9.10 x86_64 Linux distribution). These results were obtained using some efficient tricks in the implementation which we are unable to report here due to paper size limitations, but hope to describe in detail in an extended version of the paper. It is interesting to note that all neural network based matrix factorization implementations

Table 1. RMSE of models on the Probe and Quiz sets

	NNMF (800)	NNMF-S (800)	NNMF-SS (800)	RBM (100)	COND- RBM (100)	NN- AUTO-SS (50)	RBM- SOM (15 × 15)	RBM- SOM- RBF (15 × 15)
Probe	0.9026	0.9018	0.9017	0.9293	0.9204	0.9294	0.9229	0.9202
Quiz	0.8955	0.8948	0.8933	0.9257	0.9172	0.9214	0.9167	0.9148
Time	1.7 hrs	2.0 hrs	2.5 hrs	4.5 hrs	4.6 hrs	1.8 hrs	5 hrs	5.1 hrs

(NNMF, NNMF-S and NNMF-SS) are able to achieve significantly low RMSE scores with moderate computational complexity, considering the size of the Netflix prize dataset and the large architectures of the networks (number of hidden nodes). The performance of our auto-encoder network is also comparable to those achieved by the two RBMs but NN-AUTO-SS is able to deliver its good results with much smaller computational effort. Finally, it is also interesting to note that our unified hybrid approaches RBM-SOM and RBM-SOM-RBF can improve further on the good performance of the RBM and COND-RBM methods with small computational overhead.

References

1. Bennett, J., Lanning, S.: The netflix prize. In: Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)
2. The netflix prize leaderboard, <http://www.netflixprize.com/leaderboard>
3. Netflix official winner announcement: grand prize awarded to team bellkors pragmatic chaos, <http://www.netflixprize.com/community/viewtopic.php?id=1537>
4. Koren, Y.: The BellKor Solution to the Netflix Grand Prize (August 2009), http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
5. Toscher, A., Jahrer, M., Bell, R.: The Big Chaos Solution to the Netflix Grand Prize (August 2009), http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf
6. Piotte, M., Chabbert, M.: The Pragmatic Theory Solution to the Netflix Grand Prize (August 2009), http://www.netflixprize.com/assets/GrandPrize2009_BPC_PragmaticTheory.pdf
7. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: Proceedings of the International Conference on Machine Learning, vol. 24, pp. 791–798 (2007)
8. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural. Comput. 14(8), 1771–1800 (2002)

A Cooperative and Penalized Competitive Learning Approach to Gaussian Mixture Clustering

Yiu-ming Cheung and Hong Jia

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

Abstract. Competitive learning approaches with penalization or cooperation mechanism have been applied to unsupervised data clustering due to their attractive ability of automatic cluster number selection. In this paper, we further investigate the properties of different competitive strategies and propose a novel learning algorithm called Cooperative and Penalized Competitive Learning (CPCL), which implements the cooperation and penalization mechanisms simultaneously in a single competitive learning process. The integration of these two different kinds of competition mechanisms enables the CPCL to have good convergence speed, precision and robustness. Experiments on Gaussian mixture clustering are performed to investigate the proposed algorithm. The promising results demonstrate its superiority.

Keywords: Competitive Learning, Cooperation, Penalization, Gaussian Mixture Clustering, Number of Clusters.

1 Introduction

As a typical competitive learning algorithm, k-means [1] has a variety of applications in the different scientific areas such as image processing, clustering analysis, and so on. However, it suffers from a selection problem of cluster number. That is, k-means needs to pre-assign the number of clusters exactly; otherwise, it will almost always give out an incorrect clustering result.

In the literature, one main kind of technique to solve this selection problem is to introduce some competitive learning mechanisms into an algorithm so that it can perform automatic cluster number selection during the learning process. For example, the Rival Penalized Competitive Learning (RPCL) [2] can automatically select the cluster number by gradually driving extra seed points, i.e. those data points that are learnable towards the cluster centers in the input space, far away from the input data set. Nevertheless, the empirical studies have also found that the performance of RPCL may completely break down without an appropriate delearning rate. Under the circumstances, paper [3] has proposed an improved version, namely Rival Penalization Controlled Competitive Learning (RPCCL),

which determines the rival-penalized strength based on the distance between the winner and the rival relative to the current input. However, both of RPCL and RPCCL always penalized the extra seed points even if they are much far away from the input data set. Consequently, the seed points as a whole will not tend to convergence. By contrast, another variant of RPCL called Stochastic RPCL (S-RPCL) [4], developed from the Rival Penalized Expectation-Maximization (RPEM) algorithm, can lead to a convergent learning process by penalizing the nearest rival stochastically based on its posterior probability. Nevertheless, when the data clusters are overlapped, the convergence speed of S-RPCL, as well as the RPCL, may become slow and the final locations of seed points may have a bias from the cluster centers.

Alternatively, Competitive and Cooperative Learning (CCL) [5] implements a cooperative learning process, in which the winner will dynamically select several nearest competitors to form a cooperative team to adapt to the input together. The CCL can make all the seed points converge to the corresponding cluster centers and the number of those seed points stayed at different positions is exactly the cluster number. Nevertheless, the performance of CCL is somewhat sensitive to the initial positions of seed points. To overcome this difficulty, Li et al. [6] have proposed an improved variant, namely Cooperation Controlled Competitive Learning (CCCL) method, in which the learning rate of each seed point within the same cooperative team is adjusted adaptively. Unfortunately, the CCCL may still not work well if the initial seed points are all gathered in one cluster.

This paper will present a new competitive learning algorithm, namely Cooperative and Penalized Competitive Learning (CPCL), which performs the two different kinds of learning mechanisms simultaneously: cooperation and penalization, during the single competitive learning process. That is, given an input, the winner generated from the competition of all seed points will not only dynamically select several nearest competitors to form a cooperative team to adapt to the input together, but also penalize some other seed points which compete intensively with it. The cooperation mechanism here enables the closest seed points to update together and gradually converge to the corresponding cluster centers while the penalization mechanism supplies the other seed points with the opportunity to wander in the clustering space and search for more appropriate cluster centers. Consequently, this algorithm features the fast convergence speed and the robust performance against the initialization of the seed points. The experiments have demonstrated its outstanding performance and robustness.

2 Cooperative and Penalized Competitive Learning (CPCL) Algorithm

Suppose N inputs, X_1, X_2, \dots, X_N , come from k^* unknown clusters, and k ($k \geq k^*$) seed points m_1, m_2, \dots, m_k are randomly initialized in the input space. Subsequently, given an input X_i each time, as described in [7], the winner among k seed points is determined by

$$I(j|X_i) = \begin{cases} 1, & \text{if } j = \arg \min_{1 \leq r \leq k} \gamma_r \|X_i - m_r\|^2 \\ 0, & \text{otherwise} \end{cases}, \quad (1)$$

with the relative winning frequency γ_r of m_r defined as

$$\gamma_r = \frac{n_r}{\sum_{j=1}^k n_j}, \quad (2)$$

where n_j is the winning frequency of m_j in the past. That means the winning chances of frequent winning seed points are gradually reduced by an implicit penalty. After selecting out the winner m_w , the circle centered at m_w with the radius $\|m_w - X_i\|$ is regarded as the territory of m_w . Any other seed points intruded into this territory will either cooperate with the winner or be penalized by it.

The winner m_w in this learning approach always chooses the seed points nearest to it as its cooperators and the number of cooperators needed by the winner is gradually increased as the learning process repeats. Consequently, we define that in the i^{th} learning epoch of the algorithm, the winner can at most choose C_i cooperators, where $C_i = \min\{i, k-1\}$. This kind of cooperating scheme ensures that the seed points have enough opportunities to drift in the whole input space and converge smoothly. Each member in the cooperating team, denoted as m_o , will be updated by

$$m_o^{new} = m_o + \eta \frac{\|m_w - X_i\|}{\max(\|m_w - X_i\|, \|m_o - X_i\|)} (X_i - m_o), \quad (3)$$

where η is a specified positive learning rate. It can be seen that a cooperator will have a full learning rate η as $\|m_o - X_i\| \leq \|m_w - X_i\|$. Otherwise, the learning strength is gradually attenuated when the distance between the cooperator and the current input increases.

The other non-cooperating seed points in the winner's territory, denoted as m_p , will be penalized with a dynamical penalizing rate:

$$m_p^{new} = m_p - \eta \frac{\|m_w - X_i\|}{\|m_p - X_i\|} (X_i - m_p). \quad (4)$$

That is, the closer the seed point is to the input, the more penalization it will suffer from. Consequently, the CPCL algorithm can be given as follows:

Step 1: Pre-specify the number k of clusters ($k \geq k^*$), and initialize the k seed points $\{m_1, m_2, \dots, m_k\}$. Set $t = 1$, $i = 1$ and $n_j = 1$ with $j = 1, 2, \dots, k$, where t and i are the number of epochs and input data, respectively.

Step 2: Given an input X_i , calculate $I(j|X_i)$ by Eq. (1).

Step 3: Determine the winner unit m_w . Let S_w be the set of seed points fallen into the territory of m_w . That is, let $S_w = \{m_w\}$, and span S_w by

$$S_w = S_w \cup \{m_j | \|m_w - m_j\| \leq \|m_w - X_i\|\}. \quad (5)$$

Step 4: Sort the units in S_w based on the distance between each unit to the winner m_w . We denote the sorted units as: m'_1, m'_2, \dots, m'_s , with

$$\|m'_1 - m_w\| \leq \|m'_2 - m_w\| \leq \dots \leq \|m'_s - m_w\| \quad (6)$$

where $s = |S_w|$.

Step 5: Select a subset S_c of S_w to form a cooperating team of m_w , where $S_c = \{m'_1, m'_2, \dots, m'_q\}$ with $q = |S_c| = \min\{s, t\}$. Then update all members in S_c by Eq. (3).

Step 6: Let $S_p = S_w - S_c$, then, penalize all seed points in S_p by Eq. (4).

Step 7: Update n_w by $n_w^{new} = n_w^{old} + 1$. Let $i = i + 1$, $t = 1 + \lfloor i/N \rfloor$.

The above Step 2 to Step 7 are iterated for each input until all m_j 's converge.

3 Experimental Results

3.1 Experiment 1

To demonstrate the performance of the CPCL algorithm in comparison with the CCCL and S-RPCL, we generated 1,000 data points from a mixture of three 2-dimension Gaussian densities:

$$p(X|\Theta) = 0.3G(X|\mu_1, 0.15I) + 0.4G(X|\mu_2, 0.15I) + 0.3G(X|\mu_3, 0.15I), \quad (7)$$

with $\mu_1 = [1.0, 1.0]^T$, $\mu_2 = [1.0, 2.5]^T$ and $\mu_3 = [2.5, 2.5]^T$. Six seed points were randomly initialized in the input space. For each algorithm, the learning rate η was set at 0.001 and φ in CCCL algorithm was set to 0.5 according to [6].

After 200 learning epochs, the positions of seed points obtained by the three algorithms are shown in Fig. 1(a) to Fig. 1(c), respectively. It can be seen that all the three algorithms have identified the true number of clusters successfully. However, the S-RPCL had not located the cluster centers accurately yet. Moreover, Fig. 2 shows the learning curve of m_j 's via each method. It can be seen that the CPCL approach is faster than the CCCL and the S-RPCL had not converged after 200 epochs. This scenario shows the good performance of CPCL in terms of convergence rate and precision.

3.2 Experiment 2

In this experiment, we further investigated the performance of CPCL on the mixture clusters that were seriously overlapped and a cluster was in elliptical

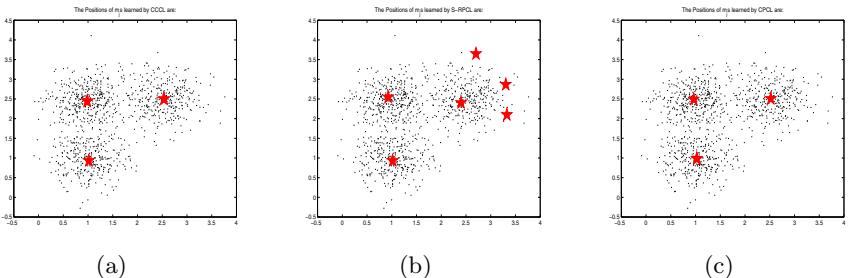


Fig. 1. The positions of six seed points marked by “★” in the input space in Experiment 1 learned by (a) CCCL, (b) S-RPCL and (c) CPCL respectively

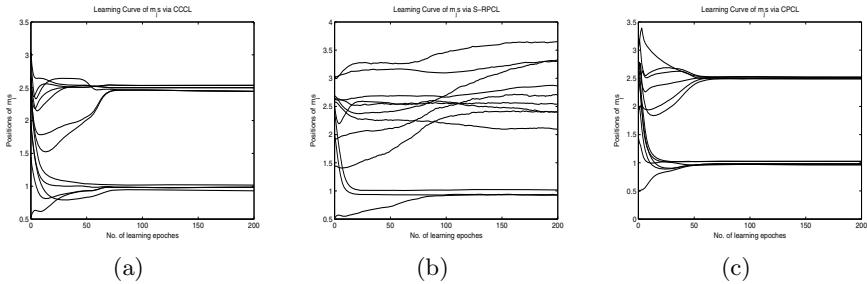


Fig. 2. The learning curves of six seed points in Experiment 1 obtained by (a) CCCL, (b) S-RPCL, and (c) CPCL, respectively

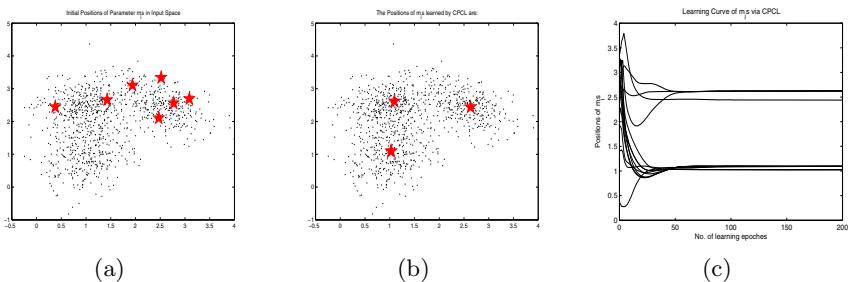


Fig. 3. The results of Experiment 2: (a) the initial positions of seven seed points marked by "★" in the input space, (b) the positions of seed points learned by CPCL, and (c) the learning curves of seven seed points

shape. Similar to Experiment 1, 1,000 data points were generated from a mixture of three 2-dimension Gaussian densities:

$$p(X|\Theta) = 0.3G(X|\mu_1, \Sigma_1) + 0.4G(X|\mu_2, \Sigma_2) + 0.3G(X|\mu_3, \Sigma_3), \quad (8)$$

with $\mu_1 = [1.0, 1.0]^T$, $\mu_2 = [1.0, 2.5]^T$, $\mu_3 = [2.5, 2.5]^T$, $\Sigma_1 = \begin{pmatrix} 0.2, 0.05 \\ 0.05, 0.3 \end{pmatrix}$, $\Sigma_2 = \begin{pmatrix} 0.2, 0.0 \\ 0.0, 0.2 \end{pmatrix}$ and $\Sigma_3 = \begin{pmatrix} 0.2, -0.1 \\ -0.1, 0.2 \end{pmatrix}$. We randomly initialized 7 seed points in the input space as shown in Fig. 3(a). From Fig. 3(b), we can see that the seed points learned by CPCL had been converged accurately to the corresponding cluster centers after 200 learning epochs. Furthermore, Fig. 3(c) shows that the learning curves of m_j s had converged during the first 50 epochs. Also, it can be seen from this experiment that, although the concept of winner's territory in CPCL is based on Euclidean distance only, this new algorithm can work well on not only ball-shaped data clusters, but also elliptical ones.

4 Concluding Remarks

In this paper, we have presented a novel CPCL algorithm, which performs the competition with the two different kinds of mechanisms simultaneously: cooperation and penalization. On the one hand, the cooperation mechanism enables the closest seed points to update together and gradually converge to the corresponding cluster centers, which gives the algorithm good convergence speed and high precision. On the other hand, the penalization mechanism provides the other seed points with the opportunity to wander in the clustering space, which enables it to perform the clustering problem with the robustness against the initialization of the seed points and the overlap of the data clusters. The experimental results have shown its outstanding performance. Essentially, the underlying learning mechanism of the proposed approach is applicable to the general density mixture clustering, although this paper has presented the details of this approach on Gaussian mixture clustering only.

Acknowledgment

The work described in this paper was supported by the Faculty Research Grant of Hong Kong Baptist University with the project code: FRG2/08-09/122, and by the Research Grant Council of Hong Kong SAR under Project HKBU 210309.

References

1. MacQueen, J.B.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, Calif., USA, pp. 281–297 (1967)
2. Xu, L., Krzyzak, A., Oja, E.: Rival Penalized Competitive Learning for Clustering Analysis. RBF Net, and Curve Detection. *IEEE Transactions on Neural Networks* 4, 636–648 (1993)
3. Cheung, Y.M.: Rival penalization controlled competitive learning for data clustering with unknown cluster number. In: Proceedings of 9th International Conference on Neural Information Processing, pp. 18–22 (2002)
4. Cheung, Y.M.: Maximum weighted likelihood via rival penalized EM for density mixture clustering with automatic model selection. *IEEE Transactions on Knowledge and Data Engineering* 17, 750–761 (2005)
5. Cheung, Y.M.: A competitive and cooperative learning approach to robust data clustering. In: Proceedings of the IASTED International Conference of Neural Networks and Computational Intelligence (NCI 2004), pp. 131–136 (2004)
6. Li, T., Pei, W.J., Wang, S.P., Cheung, Y.M.: Cooperation Controlled Competitive Learning Approach for Data Clustering. In: Proceedings of 2008 International Conference on Computational Intelligence and Security (CIS 2008), pp. 24–29 (2008)
7. Ahalt, S.C., Krishnamurty, A.K., Chen, P., Melton, D.E.: Competitive learning algorithms for vector quantization. *Neural Networks* 3, 277–291 (1990)

A Inference Mechanism for Polymer Processing Using Rough-Neuro Fuzzy Network

Carlos Affonso and Renato Jose Sassi

Industrial Engineering Post Graduation Program, Nove de Julho University, São Paulo, Brazil

Abstract. There is an increasing tendency in the worldwide automotive market to consume polymeric materials, because of their processability and low cost in high volumes. This need motivates the search for technological improvements to the material performance, even at the product development stage. The purpose of this paper is to predict the cycle time of an injected part according to its molding parameters using a Rough-Neuro Fuzzy Network. The methodology involves the application of Fuzzy Sets to define an inference mechanism that inserts human knowledge about polymer processing into a structured rule basis. The attributes of the molding parameters are described using membership functions and reduced by Fuzzy rules. The Rough Sets Theory identifies the attributes that are important and the Fuzzy relations influence the Artificial Neural Network (ANN) surface response. Thus, the rule basis filtered by Rough Sets is used to train a back-programmed Radial Basis Function (RBF) and/or a Multilayer Perceptron (MLP) Neuro Fuzzy Network. In order to measure the performance of the proposed Rough-Neuro Fuzzy Network, the responses of the unreduced rule basis are compared with the reduced rule basis. The results show that by making use of the Rough-Neuro Fuzzy Network, it is possible to reduce the need for expertise in the construction of the Fuzzy inference mechanism.

Keywords: Neuro Fuzzy Network, Polymer Processing, Rough Set.

1 Introduction

There is currently a worldwide trend in the automotive market to increase the use of polymeric materials, due to their easy processability and low cost of production, especially in large volumes [1]. However, ensuring the quality of products is complex, due to the many factors that influence this process, from product conception through to the construction of the injection mould. There are several commercial software applications for modeling the parameters of the injection of polymers using finite element methods. The predictions of this program are based on both the rheological/thermal properties and the product geometry [2].

It is possible to find these parameters analytically; however, solving this problem by applying classical theories of transport phenomena (Navier-Stokes equations) requires accurate information about the injection machine, product geometry, and process parameters. Therefore, the analysis of the injection mould process by classical methods using mathematical models is complex because of the following factors [3]:

The moulds often have complex shapes; the melt flow is transient; the thermal effects are relevant; and the melt flow occurs in a high shear rate. One important type of complex knowledge can occur on inference mechanism from multiple relations, such polymer processing. In these domains, the objects of interest are not independent of each other, and are not of a single type. We need inference mechanism that can soundly mine the rich structure of relations among objects [4].

Considering the above points, we think the use of Fuzzy Sets is necessary, because this type of approach is especially applicable to systems where information is inaccurate and the strategies used by human experts in dealing with these systems are expressed through linguistic terms [5,7].

The theory of rough sets [6] has recently emerged as another major mathematical approach for managing uncertainty that arises from inexact, noisy, or incomplete information. It is found to be particularly effective in the area of knowledge reduction.

The intention was to use rough sets as a tool for structuring the neural networks. The methodology consisted of generating rules from training examples by rough-set learning, and mapping the dependency factors of the rules into the connection weights of a four-layered neural network.

Fuzzy set theory [5] and rough set theory [6] represent two different approaches to vagueness. Fuzzy set theory addresses gradualness of knowledge, expressed by the fuzzy membership, whereas rough set theory addresses granularity of knowledge, expressed by the indiscernibility relation [8,9].

The great advantage of the Rough-Neuro Fuzzy Network approach consists in the synergy achieved by combining two or more technical capabilities to achieve a more powerful system with respect to learning and generalization [9]. A sequential architecture is used in this work, where Rough Sets and the Neuro Fuzzy Network have distinct functions: Rough Sets identify the most critical features, while the Neuro Fuzzy Network generates the surface response input / output, because the Neuro Fuzzy Network has Learneability and can adapt itself to the real world [10]. The paper is organized as follows: section 2 introduces the fundamental concepts of Rough Sets. Section 3 presents the application of Fuzzy Sets to polymer processing. Section 4 presents a proposal for an application of Rough-Neuro Fuzzy Networks to polymer processing. The methodology, experiments, and results are presented in section 5 and in section 6, the conclusion and discussion

2 Rough Set Theory

The Rough Sets Theory (RS) was proposed by Zdzislaw Pawlak in 1982 [6] as a mathematical model to represent knowledge and to treat uncertainty. An important concept in RS is the *reduct*. A reduct is a minimal set of attributes that can represent an object with the same accuracy as the original set of attributes. Elimination of redundant attributes can help in the identification of strong, non-redundant classification rules [8].

A reduct of $B - \text{RED}(B)$ – on information system (IS) is a set of attributes $B' \subseteq B$ such that all attributes $a \in (B - B')$ are dispensable. Thus, $U/\text{INDs}(B') = U/\text{INDs}(B)$, where $\text{INDs}(B)$ is called the Indiscernibility Relation.

Computing the reduct is an n-p hard problem, and processing the reduct for large databases requires high computational processing. The reduct is generated by discernibility from the Discernibility Matrix.

The Discernibility Matrix of information systems S, denoted $DM(B)$, is a symmetric $n \times n$ matrix with: $mD(i, j) = \{a \in B \mid a(Ei) \neq a(Ej)\}$ for $i, j=1, 2, \dots, n$. with $1 \leq i, j \leq n$ e $n=|U / INDs(B)|$.

Thus, the elements of the Discernibility Matrix $mD(i, j)$ are a set of conditional attributes of B that differentiate the elements of classes in relation to their nominal values.

$$F_S(\mathbf{a}_1^*, \mathbf{a}_2^*, \dots, \mathbf{a}_m^*) = \wedge \left\{ \vee m_D^*(i, j) \mid i, j = 1, 2, \dots, n, \quad m_D(i, j) \neq 0 \right\} \quad (1)$$

With: $m_D^*(i, j) = \{\mathbf{a}^* \mid \mathbf{a} \in m_D(i, j)\}$

The reducts of S are generated through the simplification methods of Boolean functions for the $F_S(B)$ function. This simplification is an algebraic approximation of the logical functions, with the goal of reducing the number of attributes.

The discernibility function $F_S(B)$ is obtained as follows: for all attributes represented by an element in the Discernibility Matrix $MD(B)$, apply the sum operator (“or” or “ \vee ”) and, for each pair of cells in this matrix, apply the “product” element (“and” or “ \wedge ”), which results in a Boolean expression of “sum of products”. Fuzzy Sets concern membership among elements from the same class, while RS concerns the relationship between groups of elements in different classes. However, the theory of RS does not compete with the Fuzzy Sets Theory but rather complements it.

In fact, RS theory and Fuzzy Sets theory are two independent approaches for the treatment of imprecise knowledge [11].

The knowledge acquisition bottleneck is a significant problem that hinders the building of intelligent monitoring systems. The generation of good knowledge bases for this task is notoriously difficult. This problem is particularly prevalent where experts are not readily available. Machine learning techniques (especially rule induction methods) can be of great benefit to this area by providing strategies to automatically extract useful knowledge, given enough historical data [12]. Rough Selection provides a means by which discrete or real-valued noisy data (or a mixture of both) can be effectively reduced without the need for user-supplied information.

Additionally, this technique can be applied to data with continuous or nominal decision attributes, and, as such, can be applied to regression as well as classification datasets. The only additional information required is in the form of Fuzzy partitions for each feature, which can be automatically derived from the data. This corresponds to the case where only the decision attribute values are Fuzzy; the conditional values are crisp [13].

3 Fuzzy Sets Applied to Polymer Processing

In 1965, Zadeh [5] assigned a number to every element in the universe, which indicates the degree (grade) to which the element belongs to a Fuzzy set. To formulate this concept of Fuzzy set mathematically, we present the following definition [14]. Let

X be the universe. A mapping $A: X \rightarrow [0,1]$ is called a Fuzzy set on X . The value $\mu(x)$ of A , at $x \in X$ stands for the degree of membership of x in A . The set of all Fuzzy sets on X will be denoted by $F(X)$. $\mu(x) = 1$ means full membership, $\mu(x) = 0$ means non-membership, and intermediate values between 0 and 1 mean partial membership. $\mu(x)$ is referred to as a membership function as x varies in X . The production of parts by injection processes involves a series of physical phenomena that affect the process. Figure 1 and table 1 show some of the variables that operate in this process.

Table 1. Injection Mould Variables

y	Cycle time	CL
x_1	Melt Flow	MF
x_2	Part Weight	PW
x_3	Polymer Temperature	PT
x_4	Mould Temperature	MT
x_5	Injection Pressure	IP
x_6	Retarded Pressure	RP
x_7	Injection Velocity	VL

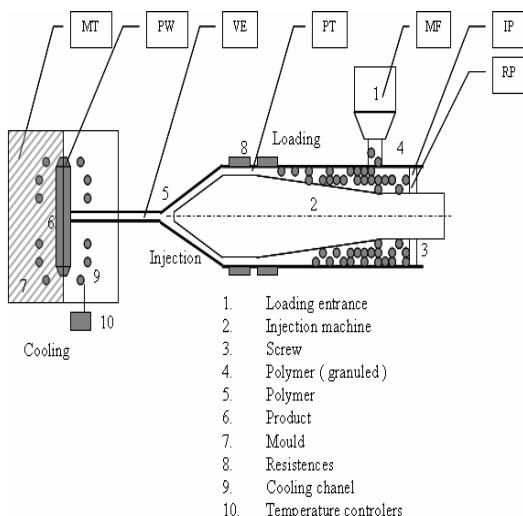


Fig. 1. Injection Mould Process

Based on the database, membership functions for all the variables involved in the injection process were defined, and the criterion for defining the deviation and center of each of these functions was developed with support from the injection experts. We considered 3 levels of sensitivity (linguistic labels) α for each membership function: low, medium, high, each with respective centers c_α and standard deviations σ .

To model the membership functions, we used the Gaussian function.

$$\mu^{\alpha}(x) = e^{-\frac{1}{\sigma}(x - c_{\alpha})^2} \quad (2)$$

Two stages are required to establish an inference mechanism to simulate a system, as suggested in [15]: The premises of all rules are compared with controlled entries to determine which rules apply to a situation; The outputs are compared with the established rules that have been determined. In this paper, we used the T-norm applied to $x = (x_1, x_2, \dots, x_n)$ as suggested by [16], where the value of $\Phi_{j, j=1, \dots, m}$ rules of inference is calculated as follows:

$$\Phi_j(x, y) = T(R_j(x), \mu_b^{\alpha_y}(y)) \quad (3)$$

where Fuzzy relations $R_{j, j=1, \dots, m}$ for each inference rule are given as:

$$R_j(x) = \mu_{x1}^{\alpha_1}(x_1) \wedge \mu_{x2}^{\alpha_2}(x_2) \wedge \dots \wedge \mu_{xn}^{\alpha_n}(x_n) \quad (4)$$

To obtain the network output value at the generalization phase, it is necessary to classify the values from inference rules [16]. Classifying the rules among learning (Φ_i^L) and generalization (Φ_j^G) phases, was achieved by minimizing the Euclidean distance, as below.

$$d_{ij} = |\Phi_i^L - \Phi_j^G| \quad (5)$$

4 A Rough-Neuro Fuzzy Network

Figure 2 illustrates the full process of the Rough-Neuro Fuzzy Network. The RS preprocesses the database and generates the input vectors with minimal attributes [17]. Then, the inference mechanism establishes the values of the rules Φ_j for the injection process, as shown in equation. This reduced rule base is used to train the RBF and MLP ANN.

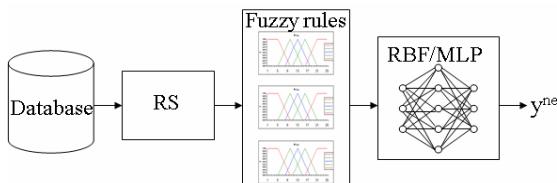


Fig. 2. A Rough-Neuro Fuzzy Network

This paper presents a supervised feedback ANN architecture in 3 layers: the input layer represents the values of the reduced rule base. The neurons of the network hidden layer are trained from the set of inference rules (reduced).

Initially, the RBF architecture will be considered. If this architecture is used to approximate the surface response, a combination of Radial Basis Functions is required. In this particular case, the Green function is required [18].

$$G(\Phi - c_i) = e^{-\frac{(\Phi - c_i)^2}{\sigma}} \quad (6)$$

The learning algorithm chosen for this application was the K-medium, where the synaptic weights W and the centers of the basis functions C are recalculated from the parameters of learning η_1 and η_2 in order to minimize the error ϵ , as follows[18]:

$$w(n+1) = w(n) + \eta_1 \sum \epsilon(n) G_{ij}(x_j - c_i) \quad (7)$$

$$c_k(n+1) = c_k(n) + \eta_2(x - c) \quad (8)$$

The ANN architecture MLP typically consists of a specification of the number of layers, the type of activation function of each unit, and the weights of connections between the different units and should be established for the construction of the neural architecture [18]. The algorithm used in training the MLP is the error backpropagation that works as follows: first, a standard is presented. In this work, a standard will be a prototype vector and its label - the input layer of the network. This pattern is processed layer by layer until the output layer provides the response rendered, f_{MLP} , calculated as shown below [18]:

$$f_{MLP}(x) = \varphi \left(\sum_{i=1}^n v_i \varphi \left(\sum_{j=1}^m w_{ij} x_j + b_{i0} \right) + b_0 \right) \quad (9)$$

Where v_i and w_{ij} are synaptic weights; b_{i0} and b_0 are the biases; φ is the activation function, usually specified as the sigmoid function.

5 Conduct of Experiments

This work should be viewed as an application of artificial intelligence techniques, in particular, ANN to determine a surface response. In accordance with this approach, Neuro Fuzzy Networks have been used to solve manufacturing problems.

The literature [1,3] show that the production of parts through polymer injection is a process where direct mathematical modeling is not feasible due to the large number of variables and physical phenomena involved. Moreover, as is typical in studies that use Fuzzy Logic, such production requires indispensable tacit knowledge of experts in the field[19]. Experiments were conducted analyzing figures from an industrial dataset, using the same inference mechanism discussed above. Thus, the data base was run with both RBF (subsection 5.1) and MLP (subsection 5.2) ANN.

The algorithms for the RBF and MLP networks have used different strategies in programming. For the MLP, subroutines available in Scilab 5.1 were used for teaching the network, according to its pre-established generalization strategies.

The hardware platform used in the experiments was a Pentium Dual Core with 2.4 MHZ, 512 MB RAM and 40 GB hard drive.

5.1 Rough-Neuro Fuzzy RBF Network Applied to Polymer Processing

The experiment compared the knowledge generated by the Rough-Neuro Fuzzy RBF Network with the data obtained in a real injection mould process in two cases: the unreduced rule base and the rule base reduced by the RS. The database was obtained from the trials in the injection branch of an auto parts industry company. The values obtained are displayed below in table 2.

Table 2. Industrial database

Try out	MF [g/10 min]	PW [g]	PT [°C]	MT [°C]	IP [bar]	RP [bar]	VL [s]	CL [s]
1	7	174	227	70	58	30	15	70,00
2	13	932	240	30	90	45	5	80,00
3	7	174	232	70	55	30	15	70,00
4	7	174	227	70	65	30	15	70,00
5	2,2	174	238	70	58	30	18	70,00
6	2,2	174	243	70	65	45	20	70,00
7	13	630	220	30	100	50	3	66,00
8	13	630	230	45	84	40	4	84,00
9	13	630	240	30	93	55	3	94,00
10	13	932	220	30	93	50	5	68,00
11	7	174	227	70	55	30	15	70,00
12	13	932	240	30	90	45	5	80,00

The cycle time was chosen as the output of this database. This parameter corresponds to the total time required to produce a part that meets quality requirements. An inference mechanism for the learning and generalization phases was established from the membership functions of each variable. The choice of RBF network parameters were: number of input neurons: 8; number of radial basis functions: 4, learning rate constant $\eta_1 = 1 \times 10^{-6}$; $\eta_2 = 9.5 \times 10^{-4}$; the stop criteria was average error $\varepsilon = 0.65$; and the maximum iterations number 2000. Figure 3 shows the output for the learning and generalization phases, comparing to the real data (experimental), the reduced rule base and the full rule base.

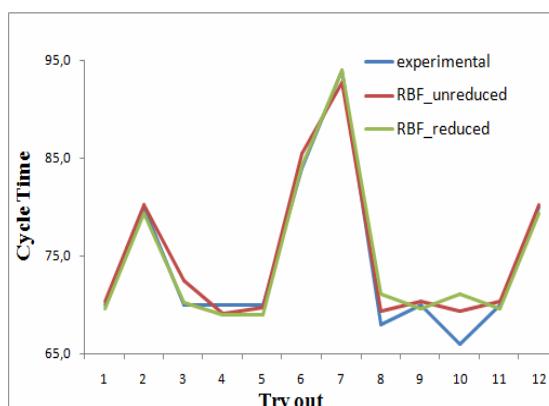


Fig. 3. RBF: Reduced and unreduced rule base-2000 epochs

5.2 A Rough-Neuro Fuzzy MLP Network Applied to Polymer Processing

The methodology discussed in the previous experiment will be repeated. The choice of parameters for the MLP was: input neurons 8, number of hidden neurons 3, and learning rate constant $\eta = 1$; the stop criteria was the average absolute error $\varepsilon = 0.85$; and the maximum interactions number 200. Figure 4 shows the output for the learning and generalization phases, comparing to the real data (experimental), the reduced rule base and the unreduced rule base, considering the cycle time injection as the output variable.

Figure 5 below shows the value of the average error, $\varepsilon = 1/n_e \sum |Y_{ANN} - Y_{REAL}|$, $n_e = n$. of events for the MLP ANN.

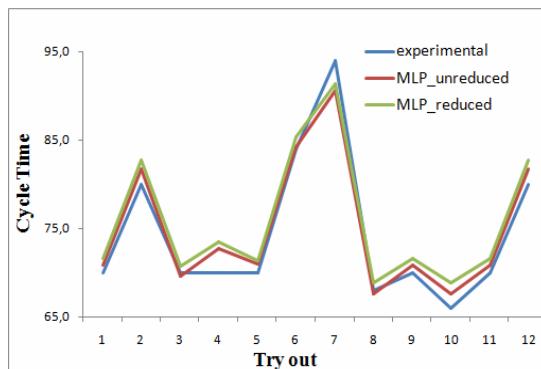


Fig. 4. MLP: Reduced and unreduced rule base-100 epochs

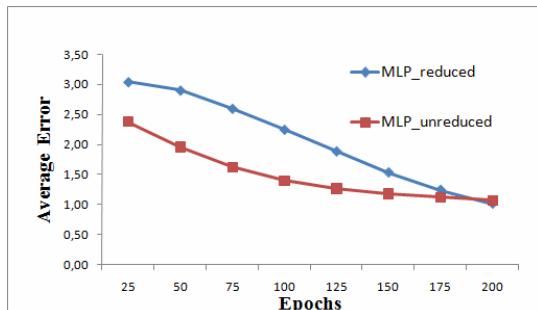


Fig. 5. Average error: MLP reduced and unreduced databases

A Rough-Neuro Fuzzy Network has a good response surface, for both RBF and MLP networks, as can be seen in figure 5, which shows the values of average error obtained. MLP networks have initially higher average error, which decreases asymptotically with the increase in the epochs. Both rule bases reduced by the RS exhibit good behavior during the generalization phase, indicating a promising way for to associate RS with Neuro Fuzzy networks to replace the human expert in the construction of inference rules [19].

6 Discussion and Conclusion

The application of Rough-Neuro Fuzzy Networks showed a great ability to generalize, identify behavior patterns, and allow the creation of an inference mechanism in high complex systems, such as injection moulding processes.

The results also permit the identification of the best setup strategies, leading to lower cycle times with the potential to optimize the use of energy resources, while improving the quality of the final product.

The RBF and MLP network architectures showed satisfactory results, in line with the experimental data extracted from the polymer processing database.

We conclude that:

- The characteristics of product and process can be correctly characterized by Fuzzy Sets.
- It is possible to create an inference mechanism for injected parts with different materials and geometries.

Furthermore, by performing a comparative analysis between the Reduced and Unreduced rule basis, it is possible to conclude:

-The reduced rule basis has a better ability to generalize that an unreduced one (at 200 epochs the average error value of the unreduced is 5.6% higher)

-The reduced rule base eliminates 71% of membership functions (from 7 to 2 membership functions for each inference rule).

-The reduced number of membership functions for each inference rule allows the creation of an automatic inference mechanism without the support of a human expert.

When applied to a real world dataset, the Rough-Neuro Fuzzy Network was able to identify the significant features as defined by a human expert. The main advantage of using the Rough-Neuro Fuzzy Network is the reduction of dependence on a human expert for the choice and construction of the rules of inference mechanism.

This gain is important, considering that one of the weaknesses of the approach using the Fuzzy Sets is its dependence on the human expert. If there is a reasonable number of attributes and a structured database, it may be even possible to eliminate the need for support from the human expert for the construction of inference rules, instead using his support only in the construction of membership functions.

We suggest the Rough-Neuro Fuzzy Network in modeling other problems, such as dynamic routing or business datasets, in order to assess the impact of this approach on the dependence of the human expert in building the inference mechanism.

References

1. He, W., Zhang, F.Y., Liu, T.I.: Development of a Fuzzy-Neuro System for Parameter Resetting of injection Molding. *Journal of Manufacturing Science* (2001)
2. Kiam, T.M., Perreira, C.N.: Estudo de caso de peça moldada pelo processo de injeção compressão para termoplásticos utilizando análise computacional. *Polímeros* 17(1) (January/March 2007)
3. Agassant, J.F., Avenas, P., Sergent, P.J., Carreau, P.J.: *Polymer Processing: Principles and Modeling, Chapter I: Continuum Mechanics*. Hanser Publishers (1991)
4. Banerjee, M., Mitra, S., Pal, S.K.: Fellow: Rough Fuzzy MLP: Knowledge Encoding and Classification. *IEEE Transactions on Neural Networks* 9(6), 1203–1216 (1998)

5. Zadeh, L.A.: Fuzzy Sets: Information and Control 8, 338–353 (1965)
6. Pawlak, Z.: Why Rough Sets? Fuzzy Systems. In: Proceedings of the Fifth IEEE International Conference on, September 8-11, vol. 2, pp. 738–741 (1996)
7. Takagi, T., Sugeno, M.: Derivation of Fuzzy control rules from human operators control action. In: Proc. IFAC Symp. Fuzzy Inform. Knowledge Representation and Decision Analysis, pp. 55–60 (1983)
8. Stepaniuk, J.: Rough – Granular Computing in Knowledge Discovery and Data Mining Rough. In: Granular Computing in Knowledge Discovery and Data Mining. Springer, Heidelberg (2008)
9. Gomide, F., Figueiredo, M., Pedrycz, W.: A neural Fuzzy network: Structure and learning. In: Bien, Z., Min, K. (eds.) Fuzzy Logic and Its Applications, Information Sciences and Intelligent Systems, pp. 177–186. Kluwer Academic Publishers, Netherlands (1998)
10. Carvalho, L.M.F.: A neuro-fuzzy system to support in the diagnostic of epileptic events using different fuzzy arithmetical operations. *Neuropsychiatria* (2008)
11. Pawlak, Z.: Rough Sets. *International Journal of Computer and Information Sciences*, 341–356 (1982)
12. Sassi, R.J., Silva, L.A., Del Moral Hernandez, E.: A Methodology using Neural Networks to Cluster Validity Discovered from a Marketing Database. In: 10th Brazilian Symposium on Neural Networks (SBRN), Salvador (2008)
13. Jensen, R.: Combining rough and Fuzzy set for feature selection. University of Edinburgh, Edinburgh (2005)
14. Wang, X., Ruan, D., Kerre, E.: Mathematics of Fuzziness, ch. 3. Springer, Heidelberg (2009)
15. Halgamuge, S.K., Glesner, M.: Neural networks in designing Fuzzy systems for real world applications. *Fuzzy Sets and Systems* 65 (1994)
16. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man Mach. Studies* 7, 1–13 (1975)
17. Yao, Y.Y.: Combination of Rough and Fuzzy Sets Based on Level Sets. In: Rough Sets and Data Mining: Analysis for Imprecise Data (1997)
18. Haykin, S.: Neural Networks: A Comprehensive Foundation. Willey & Sons, New York (1999)
19. Liu, T.I., Yang, X.M., Kalambur, G.J.: Design for Machining Using Expert System and Fuzzy Logic Approach. *ASME J. Mater. Eng. Perform.* 4(5), 599–609 (1995)

Data Mining Methods for Quality Assurance in an Environmental Monitoring Network

Ioannis N. Athanasiadis¹, Andrea-Emilio Rizzoli¹, and Daniel W. Beard²

¹ Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland
ioannis@athanasiadis.info, andrea@idsia.ch

² Environmental Protection Department, Saudi Aramco, Dhahran, Saudi Arabia
daniel.beard@aramco.com

Abstract. The paper presents a system architecture that employs data mining techniques for ensuring quality assurance in an environmental monitoring network. We investigate how data mining techniques can be incorporated in the quality assurance decision making process. As prior expert decisions are available, we demonstrate that expert knowledge can be effectively extracted and reused for reproducing human experts decisions on new data. The framework is demonstrated for the Saudi Aramco air quality monitoring network and yields trustworthy behavior on historical data. A variety of data-mining algorithms was evaluated, resulting to an average predictive accuracy of over 80%, while best models reached 90% of correct decisions.

1 Introduction

Sensor network recordings are prone to several types of failures related to noise, polarization, calibration, physical obstacles, humidity, communication latency, variability of meteorological conditions, or other network and sensor faults. The quality assurance process of environmental monitoring networks is critical for supporting both the scientific work and the dissemination of environmental conditions at operational time. Quality assurance and quality control procedure (QA/QC) deals with the major uncertainties that characterize the natural environment. Typically, QA/QC takes place off-line, on batches of measurements, processed by expert scientists. Experts identify erroneous measurements and validate sensor recordings, based on statistical indexes, boundary rules, and more importantly their experience. The process is empirical, thus it can not be fully automatized, as expert knowledge is hard to capture and depends on location, season and type of equipment.

The Saudi Aramco Air Quality Monitoring and Meteorology Network (AMMNET) is no exception to the above situation [1]. Expert scientists are routinely asked to review the quality of the recorded measurements, and restore missing or erroneous ones. The target of AMMNET is air quality, where uncertainties are driven by the atmospheric chemistry and physics, and the stochastic nature of the major air pollutant emission mechanisms (including those of anthropogenic and natural origins). Concerning photochemical pollutants (such as ozone) the emission mechanism is dynamic, accompanied by strong non-linearities in the underlying physical and chemical mechanisms, and therefore it has been always among the major challenges of air quality modeling and

forecasting. These characteristics make quality control a demanding task that occupies significant efforts of expert scientists. Automating the QA/QC process for AMMNET or similar environmental monitoring networks is of extreme importance, as it will relieve scientists from a tedious and laborious task.

Near-real time decision support and surveillance systems requires the identification of erroneous measurements while the monitored conditions still occur. Data uncertainty problems associated with environmental monitoring networks bring forth issues of measurement validation and estimation of missing or erroneous values, which are critical for taking trustworthy decisions in a timely fashion [2]. This vision also drives towards the extension of legacy environmental monitoring infrastructure with modular, flexible software interfaces that provide with inputs to environmental decision support systems for hazard identification and incident forecasting [3]. This paper investigates how the quality assurance process can be improved by adopting data mining methods under both near real time constraints and off-line using historical data from an environmental monitoring network.

2 Towards a Data-Driven Quality Assurance System

Key requirements: A semi-automated quality assurance *system* needs to be able to evaluate the quality of the incoming measurements both online, as they are recorded into the system, and offline, as they are grouped in batches. Semi-automated processing is an important factor for near-real time dissemination [4], but also for identification of hazardous events and operational decision making [3].

To achieve this, a **quality index** is attached to each measurement recording at the time it arrives. Assume some sensor captures a measurement. Then, the *system* stores it, and preprocesses it by attaching a measure of quality for this recording, i.e the quality index. To compute the quality index the *system* may use all information available: past recordings from the same sensor, past and concurrent recordings of other sensors (in the same or other locations), and their statistics (i.e minimum, maximum or average values during a given period, e.g. hourly).

The inputs of the system are all raw measurements, as recorded by the sensor network, and the output is the “quality index” of the measurement. This setting was selected to mimic the information that is available to the experts: Experts investigate raw measurements and they are able to indicate the quality of the recordings. We want our data-driven decision making model to be able to do so. The amount of information available to the expert includes the current recording of the target pollutant, but also the concurrent and historical values of other pollutant concentrations and meteorological attributes, with their statistics. By visually inspecting all these time-series an expert can assess the quality of the target pollutant concentration. By presenting past data with the correct decisions attached to a supervised learner, we can train it, using a classification algorithm, to approximate the behavior of the expert. Then, the produced decision making model can be used for future predictions. In this respect the *system* is deployed around a **decision model** that is triggered by incoming recordings and responds with their corresponding quality indices (Fig. 1). The *decision model* functions as a *transformation* function, that utilizes a set of inputs originating from the sensor network in

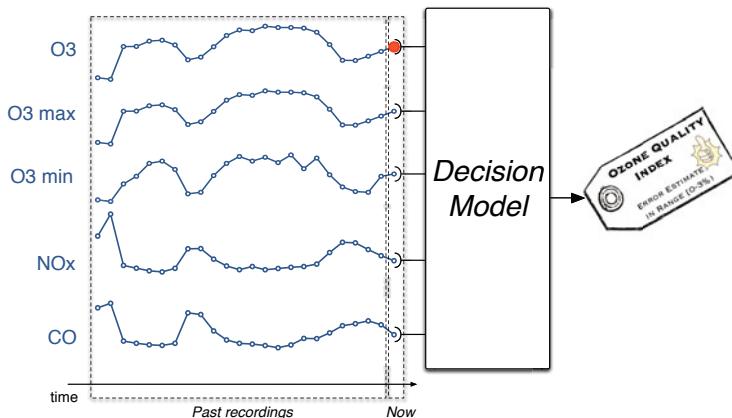


Fig. 1. The proposed abstract architecture. The decision model collects relevant time series data and responds on the quality of the currently arrived ozone measurement.

order to derive the quality index. It can be the result of a supervised learning process, i.e an expert system, a decision tree, a rule set or a neural network.

Another key aspect of the *system* is that it needs two modes of operation: one for the near-real time case and one for the off-line evaluations. In the first case, the decision model employs as inputs only concurrent and past measurements, while in the second case it may also make use of measurements recorded later.

Abstract architecture: We have designed a semi-automated quality assurance system for AMMNENET, although the architecture is not specific to the network structure or the domain, and it can be deployed for other types of monitoring networks. Figure 1 illustrates the abstract architecture of the system that serves as a mediating layer between a sensor network and any end-user applications (i.e. database, monitoring or warning systems). The system mediates in order to assign quality indices to the incoming measurements. In Fig. 1 the system responds on the quality of an ozone measurement, on the basis of past and current measurements of ozone and other pollutants, and assigns a qualitative quality index (i.e. the ozone quality index is in the range of 0-3 percent).

The quality indices and problem formulation: The quality of a recording of the monitoring network can be measured using the relative error of the raw value with respect to the correct one, as a quantitative index. In practice, we can consider the correct value to be the one that is proposed by the quality assurance expert. Let $p(t)$ be the raw value recorded at time t and $\hat{p}(t)$ the correct value, then the relative difference of the two values defines the *quantitative quality indicator*, as:

$$e_p(t) = \left| \frac{p(t) - \hat{p}(t)}{\hat{p}(t)} \right| \times 100\% \quad (1)$$

The *quantitative quality indicator* values can be classified into crisp *qualitative quality bands*, which in turn serve as a figure of merit of the raw data quality.

In the envisioned automated system, a set of thresholds $\{s_1 < s_2 < \dots < s_t\}$ will be used for defining the quality index for pollutant p as:

$$QI_p(t) = \begin{cases} \text{In range } [0, s_1) & \text{if } e_p(t) \leq s_1 \\ \text{In range } [s_1, s_2) & \text{if } s_1 < e_p(t) \leq s_2 \\ \dots \\ \text{In range } [s_t, +\infty) & \text{if } e_p(t) > s_t \end{cases} \quad (2)$$

Each range can be associated with some textual information. For example, ‘credible’ (in range:0–5%), ‘suspicious’(in range: 5–15%), and ‘distorted’ (in range: >15%). The discretization of the quality index into bands with an associated labels allows for a qualitative evaluation of raw measurements. This is particularly interesting for an online system that is able to provide with an automatically computed quality index of the incoming recordings as they arrive. The selection of the appropriate thresholds for the *Quality Index* is a matter of design choice¹. More importantly, the use of categories as a quality index instead of the real error, allows to reformulate the problem from a function approximation to a classification problem: Instead of approximating the error value, we aim to design a system that approximates the error range.

3 System Deployment: The Case of AMMNET

Design choices: Deploying a data-driven decision making system for AMMNET involves three important design choices. First comes the selection of the appropriate thresholds to define quality bands and indices. The latter serve as the outputs of the decision model. Second is the selection of the appropriate inputs for the decision model. And finally, the selection of suitable algorithms to implement the decision model.

To evaluate the feasibility of the approach, we conducted three rounds of experiments: We started with an **explorative round**, where several alternative system formulations and algorithms were screened, in order to valuate their potential. Secondly, we performed an **extensive evaluation round**, where selected algorithms and system configurations have been thoroughly studied in order to conclude to a statistically credible performance evaluation. Finally, the study concluded with a **blind-test round**, where the decision models induced in the previous round were used for newly collected data, not evaluated yet by experts and the results were presented to experts. The results of our experiments are summarized below in Section 4.

Data available: In our evaluations we focused on the quality assurance of **ozone**, and the goal was to build a data-mining model for the **ozone quality index**. We used historical data from three stations of the AMMNET network (namely Rahimah, Riyadh and Dhahran) covering in the period from January 2005 through June 2007. In each station data from several gases are monitored (namely: O_3 , H_2S , SO_2 , NO , NO_x , NO_2 , CO), along with meteorological attributes (namely: Dew Point, Precipitation, Pressure, Solar Radiation, Temperature (at 2m and 10m), Wind Speed and Direction). For all measurements both raw and quality assured data were available, at hourly intervals accompanied

¹ It can also be defined by legislation, but still a set of arbitrary thresholds.

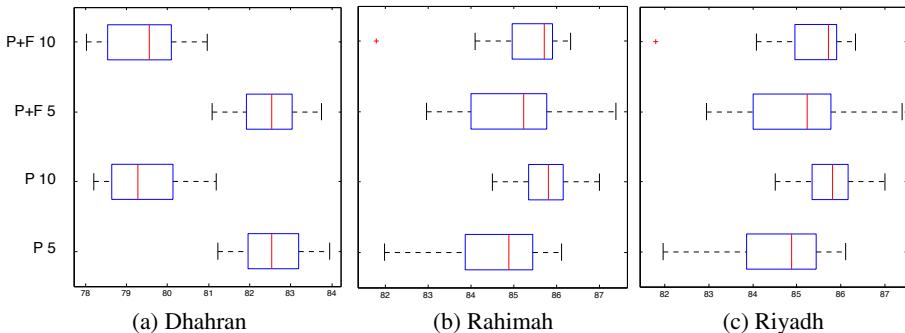


Fig. 2. Classification accuracy for all 25 training schemes, using 10-cross fold validation. Both hypotheses have been tested (marked as ‘P’ and ‘P+F’ for ‘past’ and ‘past and future’ data used as inputs), with two single quality thresholds set at 5% and 10%.

with sensory statistics (minimum value, maximum value, standard deviation). In our experiments, we used as inputs raw values.

4 Results

Explorative round of experiments: The first part of this study concentrated on the rough evaluation of alternative algorithms and inputs. We investigated a set of four alternative quality indices, 14 model inputs and 12 algorithms (with 161 configurations). In total, 27'048 evaluations were performed. In each one of them, a decision-model was trained with data spanning from January 2005 to December 2006, and the induced decision models were validate against human decision for a period covering January to June 2007. Results for Ryadh and Dhahran were very promising, for the cases where one or two quality thresholds were applied: the average performance was above 80% and the best achieved result were in the range of 95% of correct decisions. In terms of algorithm selection, most credible results were achieved by C4.5 decision trees [5], Bayesian Networks [6], Fuzzy Lattice Reasoning [7], and Multi Layer Perceptrons, implemented with WEKA [8].

Extensive evaluation round: In a second set of experiments, we evaluated thoroughly a set of algorithms in a statistically coherent way, to guard results against data partitioning biases, by applying the **10-fold cross validation** method. Both hypotheses were tested, considering a single quality threshold for ozone set at 5% or at 10%. As inputs we considered gas concentrations and meteorological attributes, accompanied by sensory statistics. We employed C4.5 decision trees with confidence factor pruning (9 configurations) and reduced error pruning (10 configurations), and bayesian networks with a simple hill climbing search algorithm (6 configurations). The classification accuracy for all training configurations are presented in Fig. 4. While the average prediction accuracy ranges above 85% for Dhahran and Riyadh, it is limited to below 75% for Rahimah. However, Rahimah’s best results outperform those of the other two stations.

Blind test results: The third round of experiments has evaluated further the best models trained above in a blind test, inspired from the Turing test: We presented a (previously unknown to the system) data stream to the decision models, and recorded model decisions. Then, model decisions have been presented to AMMNET experts, in order to analyze them and compare them with the in-house QA/QC process. Experts' response for Rahimah was very positive, while this was not the case for Riyadh.

5 Discussion and Future Work

This study concluded that data-mining algorithms selected and tuned during our experiments have yielded credible performance, and are capable for inclusion in a semi-automated system for quality assurance of a sensor network. However, expert involvement is still needed for model training and selection.

Time and space is a very important factor. The length of calibration period needs further investigation. Similarly holds for the proliferation of the extracted decision models, with respect to local incidents (i.e instrument calibration, scheduled maintenance, etc) and seasonal patterns that occur in each station. Influence across stations (spatial interactions) need further study. Another important issue is the *preferred type of failures*. Any automated system will make wrong decisions. The issue here is what type of wrong decisions we prefer. A stringent system that falsely rejects measurements is certainly preferred against an 'easy' system that falsely accepts wrong measurements.

Finally, semi-automation still remains the goal of a future course of action. An interactive *system* can be designed and deployed, so that it is capable of allowing experts to mark interesting cases that need to be considered, and to reject trivial or false patterns discovered. Such an approach will put the foundations for a win-win cycle, where experts instruct the *system* and review models extracted from data.

References

1. Beard, D.: Saudi Aramco Real-Time Air Quality and Meteorological Monitoring Network. In: Information Technologies in Environmental Engineering (ITEE), pp. 630–643. Shaker, Aachen (2005)
2. Athanasiadis, I.N., Mitkas, P.A.: Knowledge discovery for operational decision support in air quality management. Journal of Environmental Informatics 9, 100–107 (2007)
3. Athanasiadis, I.N., Milis, M., Mitkas, P.A., Michaelides, S.C.: A multi-agent system for meteorological radar data management and decision support. Environmental Modelling & Software 24, 1264–1273 (2009)
4. Athanasiadis, I.N., Mitkas, P.A.: An agent-based intelligent environmental monitoring system. Management of Environmental Quality 15, 238–249 (2004)
5. Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
6. Pearl, J.: Probabilistic reasoning in intelligent systems. Morgan Kaufmann, San Francisco (1988)
7. Kaburlasos, V.G., Athanasiadis, I.N., Mitkas, P.A.: Fuzzy Lattice Reasoning (FLR) classifier and its application for ambient ozone estimation. International Journal of Approximate Reasoning 45, 152–188 (2007)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11, 10–18 (2009)

Predicting QoL Parameters for the Atmospheric Environment in Athens, Greece

Ioannis Kyriakidis^{1,2}, Kostas Karatzas³, and George Papadourakis¹

¹ University of Glamorgan, School of Computing,
Pontypridd, Wales, United Kingdom CF37 1DL

kyriakidis@teicrete.gr

² Department of Applied Informatics & Multimedia, T.E.I. of Crete,
Estavromenos GR-71004 Heraklion, Crete, Greece

papadour@cs.teicrete.gr

³ Department of Mechanical Engineering, Aristotle University,
GR-54151 Thessaloniki, Greece
kkara@eng.auth.gr

Abstract. Air quality has a direct impact on the quality of life and on the general environment. Understanding and managing urban air quality is a suitable problem domain for the application of artificial intelligence (AI) methods towards knowledge discovery for the purposes of modeling and forecasting. In the present paper Artificial Neural Networks are supplemented by a set of mathematical tools including statistical analysis and Fast Fourier Transformations for the investigation and forecasting of hourly benzene concentrations and highest daily 8 hour mean of (8-HRA) ozone concentrations for two locations in Athens, Greece. The methodology is tested for its forecasting ability. Results verify the approach that has been applied, and the ability to analyze and model the specific knowledge domain and to forecast key parameters that provide direct input to the environmental decision making process.

Keywords: air quality, artificial neural networks, principal component analysis, Fourier transformations, ozone forecasting, benzene forecasting.

1 Introduction

Air pollution is responsible for major health impacts to humans [1]. Many different chemicals in the air affect people and the environment in adverse ways. The intensity of the risk depends on the chemicals they are exposed to, their concentration levels, and the duration of the exposure, while it may be higher for sensitive parts of the population. The present paper is focused on forecasting of: (a) hourly pollution episodes associated to Benzene (part 1), and (b) the highest daily 8-hour running average (8-HRA) of Ozone concentration levels (part 2). Benzene is a substance with carcinogenic action, and is regulated in Europe on the basis of yearly values (not to exceed 5 $\mu\text{g}/\text{m}^3$), while there are cases where it is regulated also on an hourly basis (like, i.e. the 30 $\mu\text{g}/\text{m}^3$ limit value in Alberta, Canada). Ozone is a photochemical pollutant with adverse effects in the respiratory and cardiovascular system, with a limit value for the

8-HRA levels of $120 \mu\text{g}/\text{m}^3$ [2]. Both pollutants are associated with car circulation (Benzene being directly emitted and Ozone being a secondary pollutant formulated with the synergy of other car-emitted pollutants), and are found in large cities with high traffic conditions [3]. Data from two monitoring stations (Patision and Liosia) from Athens, Greece are used (freely available via the Ministry of Environment, www.ypeka.gr). Two chronologically different datasets are employed per station, in order to evaluate the forecasting ability of our model.

The forecasting of air pollutant concentration levels was based on Artificial Neural Networks (ANNs) as a Computational Intelligence (CI) method. ANNs are computational data structures that try to mimic the biological processes of the human brain and the nervous system. ANNs were advanced in the late '80s, popularizing non-linear regression techniques like Multi-layer Perceptrons (MLP) [4] and self-organising maps (SOM) [5]. It has been proved that ANNs can be trained successfully to approximate virtually any smooth, measurable function [6]. Their advantages also include greater fault tolerance, robustness, and adaptability, especially compared to expert systems, due to the large number of interconnected processing elements that can be trained to learn from new patterns [7]. These features provide ANNs with the potential to model complex non-linear phenomenon like air pollution [8-12].

2 Materials and Methods

2.1 Data Presentation

Air quality data used come from two monitoring stations (Patision and Liosia) in Athens, Greece. Data from Patision (a city centre, traffic oriented station) are used in both parts (Benzene and Ozone forecasting) of our work. Data from Liosia (a sub-urban-background station) are used only for the second part of our work (Ozone forecasting). Two chronologically different datasets are employed per monitoring station, to evaluate the generalization properties of our model. For Benzene, at part 1 of the calculations, two datasets are used: (i) A 2-year long hourly air pollutant concentrations and meteorological information for the years 2004 and 2005 from Patision. (ii) A dataset for the years 2006 and 2007 was used from the same monitoring station for evaluation purposes. The investigation parameters from those datasets were: Benzene, CO, SO_2 , Relative Humidity, Temperature, Wind Speed and Wind Direction. For Ozone, at part 2 of the calculations, four datasets are used in total: (i) A 4-year long dataset from Liosia for the time period 2002-2005 and also (ii) a dataset for the year 2007 for evolution purposes (data for the year 2006 not being available). From Patision: (iii) a 3-year long dataset was used for the years 2002, 2003 and 2005 and (iv) an extra dataset for evaluation purposes for the year 2007 (again, data for the year 2006 not being available). The investigation parameters from those datasets were: O_3 , NO_2 , SO_2 , NO, Relative Humidity, Temperature, Wind Speed, Wind Speed at 9:00am and Wind Direction.

2.2 Data Preparation (Preprocessing)

All datasets had missing values that were automatically flagged with the value -9999. The criterion applied for excluding a whole day (i.e. 24 hours) from the data set was

the percentage of the total missing values (more than 30%). In order to calculate the missing values of the remaining days, we applied the Lagrange interpolating polynomial. It should be mentioned that for wind direction the sine and cosine transformation functions were employed ($WD=1+\sin(\theta+\pi/4)$), as they were found to lead to better results [8].

In order to perform dimensionality reduction without significant loss of information [13], Principal Component Analysis (PCA) was used at the 1st part of the calculations. By using the Kaiser Rule for component selection [14], we identified that the WD feature maybe eliminated. The remaining features accounted for 99,95% of the variation of the (i) dataset and 95,9% of the variation of the (ii) dataset. A total of five input features were used, Benzene being the target parameter.

The calculations performed in the 2nd part of the paper required further preparation in terms of parameter pre-processing. For each day of the available dataset, the highest 8-hour mean running average (HRA) of ozone was calculated, as the purpose of this study was to forecast this specific parameter. Moreover, the mean value, the max value and the highest 8 hour mean values were calculated for all the remaining parameters (with the exception of the wind speed at 9:00 am), resulting in $7 \times 3 + 1 = 22$ input parameters per day per station, for the prediction models.

2.3 Periodicity Analysis

One of the questions posed in the case of environmental pollution is whether there is any periodicity in the studied system. In order to perform periodicity estimation, the Fast Fourier Transformation (FFT) [15] was utilised. On this basis, we constructed periodograms for each one of the parameters in the data set under investigation. The highest values of strength in the periodogram, determines the periodicity of the time series. Table 1 presents the most significant strength values and their corresponding periodicities from part 1 (Benzene concentration estimation, Patision station). These findings support the hypothesis that the basic source of Benzene is traffic: this is indicated from the periodicity of traffic per week (7-days: highest traffic on working days and less on weekends). The calculation of the covariance of hourly Benzene concentrations in relation to the other parameters concludes that the highest covariance to Benzene is found for Sulphur Dioxide (SO_2), Relative Humidity (RH) and Carbon Monoxide (CO). This reinforces the hypothesis that traffic is the basic source of Benzene, especially because CO is a traffic related pollutant. Overall, Benzene hourly concentrations seem to follow the life cycle of the city operation, as the latter is being dictated by mobility patterns. Taking this into account, it is evident that population exposure is expected to be non negligible.

It's clear from Table 2 that in both monitoring stations there is a difference between the periodicities of the two datasets. The periodicity of 175-days is very pronounced in the 2007 data for Patision, indicating a shift in the behaviour of the pollutant between the cold and the worm period of the year. It also worth noting that for both locations, there is a strong 24 day periodicity, indicating a monthly cycle of the phenomenon.

Table 1. The most significant strength values and their corresponding periodicities for Benzene for Patision station

C/N	Hourly Benzene Patision Data (04-05)			Hourly Benzene Patision Data (06-07)		
	Strength	Cycles /Hour	Periodicity	Strength	Cycles / Hour	Periodicity
1	$2,36 \times 10^7$	0,0059844	7 days	$1,11 \times 10^7$	0,0061	6,8 days
2	$1,92 \times 10^7$	0,125	8 hours	$9,14 \times 10^6$	0,0066	6,3 days
3	$1,35 \times 10^7$	0,041667	24 hours	$8,74 \times 10^6$	0,0067	6,2 days
4	$8,30 \times 10^6$	0,16667	6 hours	$8,36 \times 10^6$	0,0069	6,1 days

Table 2. The most significant strength values and their corresponding periodicities for daily ozone 8-HRA of Patision and Liosia monitoring stations

Station	C/N	8-HRA Ozone (datasets (i) and (iii))			8-HRA Ozone (datasets (ii) and (iv))		
		Strength	Cycles /Day	Periodicity	Strength	Cycles /Day	Periodicity
Patision	1	$7,71 \times 10^6$	0,0035	286 days	$4,40 \times 10^8$	0,0057	175 days
	2	$2,92 \times 10^6$	0,0255	39 days	$3,02 \times 10^8$	0,0419	24 days
	3	$1,64 \times 10^6$	0,029	34 days	$2,00 \times 10^8$	0,0421	24 days
	4	$1,27 \times 10^6$	0,1485	1 days	$1,85 \times 10^8$	0,0418	24 days
Liosia	1	$9,47 \times 10^7$	0,0033	303 days	$4,48 \times 10^9$	0,0418	24 days
	2	$3,21 \times 10^7$	0,0041	244 days	$3,08 \times 10^9$	0,042	24 days
	3	$1,70 \times 10^7$	0,0025	400 days	$1,05 \times 10^9$	0,0417	24 days
	4	$1,24 \times 10^7$	0,0074	135 days	$2,38 \times 10^8$	0,013	77 days

3 Artificial Neural Networks (ANNs)

In order to model and forecast target concentrations, ANNs were applied. For this purpose, the type of the neural network to be utilized was the multilayer perceptron using the backpropagation training algorithm, due to its successful performance in similar air quality related applications. This algorithm was implemented by using the Matlab Neural Network Toolbox. Different network configurations were implemented and tested during the training phase in order to avoid local minima [16].

Data to be imported to the ANN were firstly normalized, by applying the hyperbolic tangent sigmoid transfer function, which was also applied on the hidden layers. On the output layer the linear transfer function was used. This is a common structure for function approximation (or regression) problems [17], but has also shown good results in similar studies [18]. For the training phase the Levenberg-Marquardt Back-propagation algorithm [19] is implemented.

3.1 Forecasting Hourly Benzene Values in Athens with the Aid of ANNs

In order to apply the back propagation algorithm, data were divided into three subsets, for training, validation and testing. Two distinct methods were used to separate the data for the year period 2004-2005. In the first one, the data were divided successively. For every four hourly values, two are used for training, one for validation and

one for testing. In the second method, the first 50% of the data is used for training, the next 25% for validation and finally the last 25% for testing.

After the development of the ANNs models, a set of statistical indicators was applied to provide with a numerical description of the goodness of fit, as proposed by Willmot (1982, 1985) [20]. In addition, the maximum allowed benzene concentration levels of 5 $\mu\text{g}/\text{m}^3$ and 10 $\mu\text{g}/\text{m}^3$ were used as forecasting targets (limit values). It should be noted that these values were selected on the basis of the EU limit values currently posed by EU [2] for the mean annual concentration (brief 2000/69), due to absence of a relevant criterion for the hourly concentration level. In summary, the successively data division method proved to have a less reliable forecasting ability in comparison to the second (data division) method, which was thus selected to be used for the rest of the work performed in this paper.

Twelve network configurations were implemented and tested during the training for all Benzene datasets. Table 3 presents the performance results of the different network configurations for the two different datasets. The first dataset was for the year period 2004-2005 and the second for the years 2006 and 2007.

Table 3. Performance results table of hourly benzene concentrations for two different datasets

Patisia station	Hidden Layers (\mathcal{F})			Data (04-05)			Data (06-07)		
	One	Two	Three	R	R^2	IA	R	R^2	IA
Config. 1	5	-	-	0.764	0.58	0.76	0.78	0.61	0.72
Config. 2	5	5	-	0.775	0.60	0.77	0.80	0.64	0.76
Config. 3	5	5	5	0.777	0.60	0.77	0.79	0.63	0.76
Config. 4	10	-	-	0.776	0.60	0.77	0.80	0.63	0.76
Config. 5	10	10	-	0.78	0.61	0.78	0.79	0.62	0.75
Config. 6	10	10	10	0.78	0.61	0.78	0.80	0.64	0.76
Config. 7	15	-	-	0.785	0.62	0.78	0.78	0.61	0.73
Config. 8	15	15	-	0.777	0.60	0.76	0.80	0.64	0.77
Config. 9	15	15	15	0.778	0.61	0.78	0.79	0.63	0.76
Config. 10	20	-	-	0.785	0.62	0.78	0.78	0.61	0.72
Config. 11	20	20	-	0.78	0.61	0.78	0.79	0.62	0.75
Config. 12	20	20	20	0.779	0.61	0.78	0.79	0.63	0.77

Table 4. Performance results of different network configurations for the two different datasets of Patision stations for daily 8-HRA values of Ozone

Station	Config.	Hidden Layers			Data (02,03,05)			Data (2007)			Differences	
		1	2	3	R	R^2	IA	R	R^2	IA	R	R^2
Patision	1	22	-	-	0.86	0.73	0.84	0.81	0.65	0.88	0.05	0.08
	2	22	10	-	0.85	0.73	0.84	0.80	0.65	0.84	0.05	0.08
	3	22	10	5	0.87	0.76	0.83	0.83	0.68	0.86	0.04	0.08
	4	22	22	-	0.85	0.72	0.88	0.80	0.64	0.85	0.05	0.08
	5	22	22	22	0.84	0.70	0.86	0.82	0.67	0.85	0.02	0.03
Liosia	1	22	-	-	0.86	0.73	0.87	0.73	0.53	0.83	0.13	0.2
	2	22	10	-	0.88	0.77	0.88	0.77	0.59	0.84	0.11	0.18
	3	22	10	5	0.86	0.74	0.85	0.78	0.61	0.85	0.08	0.13
	4	22	22	-	0.85	0.72	0.85	0.76	0.57	0.84	0.09	0.15
	5	22	22	22	0.88	0.77	0.89	0.72	0.51	0.81	0.16	0.26

3.2 Forecasting Daily 8-HRA Values of Ozone in Athens, with the Aid of ANNs

The initial configuration used, included 22 input parameters, leading to 22 neurons in the input layer and one neuron in the output layer per ANN model, since the objective was to forecast one characteristic (8-HRA for Patision and Liosia stations). Several (a total of 5) network configurations were implemented and tested during the training phase in order to avoid local minima [16]. The number of neurons per hidden layer was not further increased, because of the high delay during the training phase.

4 Results and Discussion

From Table 3 it is evident that the configurations 7 & 10, and configuration 8, have the best results for datasets (04-05) and (06-07) respectively. Our methodology achieved better results for the dataset (06-07) with less variance (95,9%) than the dataset (04-05) with 99,95% variance. From that outcome in can be concluded that our methodology provides good forecasting performance for hourly Benzene concentrations for the Patision monitoring station and has a high forecasting ability. From Table 4 it is clear that the forecasting performance for daily 8-HRA values of Ozone, is lower when using data for only one year (2007). This indicates that our methodology requires more than one year of data to gain better results. However it must be mentioned that the differences between the best forecasting performance of the two datasets of Patision station is very low (0,04). This concludes that even with only one year of data, our methodology can provide almost the same forecasting performance with three years of data when used to forecast daily ozone 8-HRA for Patision monitoring station. It should be also indicated that our methodology has a good forecasting ability for that station and pollutant regardless of the studied time period. This does not apply to Liosia monitoring station, because of high difference (0,10) between the best forecasting performance of the two datasets. Overall, the methodology applied for the data analysis and for the forecasting of parameters that directly affect the quality of life (Benzene and Ozone concentration levels) was proven to be successful, leading to good quality results. The models developed are able to predict these parameters with acceptable accuracy.

References

1. Brunekreef, B., Holgate, S.T.: Air pollution and health. *The Lancet* 360, 1233–1242 (2002)
2. European Environment Agency and World Health Organisation, Air and Health - Local authorities, health and environment (1999),
<http://reports.eea.europa.eu/2599XXX/en/page008.html>
3. Athanasiadis, I.N., Karatzas, K.D., Mitkas, P.A.: Classification techniques for air quality forecasting. In: BESAI 2006 Workshop on Binding Environmental Sciences and Artificial Intelligence, Part of the 17th European Conference on Artificial Intelligence (EAI 2006), August 28-September 1. Riva del Garda, Italy (2006)
4. Gallant, S.I.: Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks* 1(2), 179–191 (1990)
5. Kohonen, T.: Self-Organizing Maps, Series in Information Sciences, 2nd edn., vol. 30. Springer, Heidelberg (1997)

6. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 359–366 (1989)
7. Lippman, R.P.: An introduction to computing with neural nets. *IEEE ASSP Magazine* (1987)
8. Slini, T., Karatzas, K., Mousiopoulos, N.: Correlation of air pollution and meteorological data using neural networks. *Int. J. Environment and Pollution* 20(1-6), 218–229 (2003)
9. Chelani, A.B., Gajghate, D.G., Hasan, M.Z.: Prediction of ambient PM10 and toxic metals using artificial neural networks. *J. of Air and Waste Management Ass.* 52, 805–810 (2002)
10. Comrie, A.C.: Comparing neural networks and regression models for ozone forecasting. *J. of Air and Waste Management Ass.* 47, 653–663 (1997)
11. Kolehmainen, M., Martikainen, H., Ruuskanen, J.: Neural networks and periodic components used in air quality forecasting. *Atmospheric Environment* 35, 815–825 (2001)
12. Perez, P., Trier, A.: Prediction of NO and NO₂ concentrations near a street with heavy traffic in Santiago, Chile. *Atmospheric Environment* 35(21), 1783–1789 (2001)
13. Smith, L.I.: A tutorial on Principal Components Analysis (February 26, 2002)
14. Lance, C., Butts, M., Michels, L.: The sources of four commonly reported cutoff criteria: What did they really say? *Organizational Research Methods* 9, 202–220 (2006)
15. Rader, C.M., Brenner, N.M.: A new principle for fast Fourier transformation. *IEEE Trans. Acoust. Speech Signal Process.* ASSP-24, 264–265 (1976)
16. StatSoft, Inc. © Copyright, Neural Networks (1984-2003),
<http://www.statsoft.com/textbook/stneunet.html>
17. Matlab Help. Keywords: Neural Network Toolbox
18. Slini, T., Karatzas, K., Mousiopoulos, N.: Correlation of air pollution and meteorological data using neural networks. *Int. J. Environment and Pollution* 20(1-6), 218–229 (2004)
19. Saini, L.M., Soni, M.K.: Artificial neural network based peak load forecasting using Levenberg-Marquardt and quasi-Newton methods. *IEE Proceedings of Generation, Transmission and Distribution* 149(5), 578–584 (2002)
20. Willmott, C.J., Ackleson, S.G., Davis, R.E., Feddema, J.J., Klink, K.M., Legates, D.R., O'Donnell, J., Rowe, C.M.: Statistics for the Evaluation and Comparison of Models. *Geophys. J. (Res.)* 90(C5), 8995–9005 (1985)

Investigating Pollen Data with the Aid of Fuzzy Methods

Nikolaos Mitrakis¹, Kostas Karatzas¹, and Siegfried Jaeger²

¹ Department of Mechanical Engineering, Aristotle University,
GR-54124 Thessaloniki, Greece

kkara@eng.auth.gr

² Medizinische Universität Wien, Universitätsklinik für Hals,
Nasen und Ohrenkrankheiten, 1090 Wien, Währinger Gürtel 18-20, Austria
siegfried.jaeger@meduniwien.ac.at

Abstract. The analysis of pollen data coming from a large number of monitoring sites and related to various pollen types is a multivariate knowledge discovery problem. The relationships between the variables and the forecasting of their behaviour may successfully be performed with the aid of computational intelligence methods. The present paper deals with data coming from 25 monitoring sites in Austria, including time series of pollen counts for 10 pollen types. Fuzzy rules and fuzzy clustering were employed, as well as appropriate graphical representation, for the analysis of the behaviour of the pollen types. Results indicate the ability to extract knowledge and to forecast pollen count levels that directly affect the quality of life of allergic people.

Keywords: pollen, taxa, fuzzy clustering, fuzzy rules, computational intelligence, knowledge extraction, forecasting.

1 Introduction

The impact of airborne pollen to human health and to quality of life is important, especially for those suffering from allergies. On this basis, it is of major importance to investigate, understand and analyse the behaviour of aeroallergens. The application of Computational Intelligence (CI) methods for analyzing and modelling pollen concentration data has increased in the recent years, since it was identified that methods such as Artificial Neural Networks and Neuro-Fuzzy models, clearly outperform traditional linear methods in forecasting tasks [1], [2], [3]. Most of these applications have taken into account daily average pollen concentrations and meteorological parameters, aiming at forecasting pollen concentration of certain taxon one day ahead. The present paper focuses on yearly pollen data, with the aim to extract relationships between their count levels, seasonality and time development of their overall concentrations.

2 Data Presentation

The data used in this paper are of aggregated nature, and consist of 3917 pollen count records, each containing data for 6 up to 10 different types of pollen particles (pollen

types or taxa), originating from 25 different stations and for the years 1980 till 2008, in Austria. Table 1 presents the available attributes for each record.

Our preliminary objective was to apply a “blind test” approach, meaning that we wanted to investigate whether there may be a relationship between these attributes and the pollen types (taxa), without applying any additional knowledge from the pollen domain. It should be noted that 34 out of 3917 pollen counts have been excluded since they present outlier behaviour with large values of total and peak value; hence, we ended up with 3883 pollen count records.

Table 1. Attributes available for further analysis, for each one of the 10 pollen types

Attribute	Explanation
Particle type	One of pollen types: ○ URTI, ✕ SECA, + POAC, * PLAT, □ FRAX, ♦ CORY, ▽ BETU, △ ARTE, ◀ AMBR, ▷ ALNU
start (num)	number of the start day
end (num)	number of the end day
peak day (num)	number of the peak day
peak value	value on peak day
total	annual total catch
#days	number of days with pollen count >0
#days>=15	number of days with pollen count >15
ppp start (num)	Day when 5% of [total] has been reached
ppp end (num)	Day when 95% of [total] has been reached

3 Materials and Methods

As a first step in the analysis of the available records, the final dataset was plotted in order to have a visual overview of the behaviour of the attributes (Fig. 1a - 1f).

Fig. 1a suggests that there are pollen types demonstrating a considerable time delay between the day of their first appearance and the day when their maximum concentration was measured (like URTI and ARTE), while there are others where this time period is short (like BETU and AMBR). This means that the speed of blooming, pollen production, and release, differs between various pollen types. On the other hand, Fig. 1b reveals that there are taxa like ALNU, where their overall pollen production is reached in a specific period of the year, corresponding approx to day 150 (meaning end of May), for most monitoring sites. This is supported by the fact that *Alnus viridis* at timber line in the mountain ranges (majority of data records) flowers in May/June, while *Alnus incana* and *Alnus glutinosa* in the lowlands flower from January to March/April, depending on the weather. Other taxa reach their yearly total earlier in the year (~ day 120 for BETU), and some other later in the year (day 230 for URTI). Fig. 1c suggests that there might be pollen taxa for which their maximum overall pollen production may depend on the period of the year where they start to release their first pollen grains. Thus, for example, for AMBR the max value is reached if the start day is between day ~220 and day ~230 (first half of August).

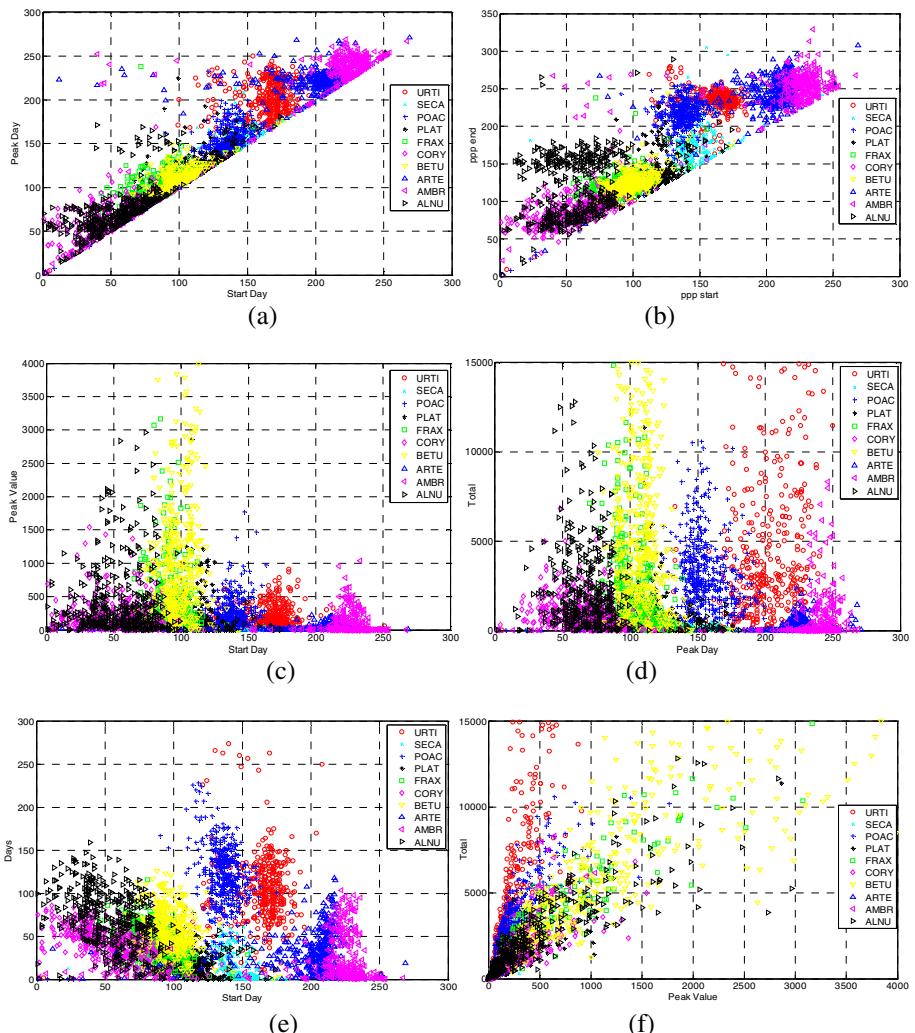


Fig. 1. Start day plotted against (a) peak day; (c) peak value; (e) number of days with non zero values. (b) day when 5% of total has reached (PPP start), plotted against day when 95% of total has reached (PPP end). (d) peak day plotted against total sum.(f) peak value plotted against total sum.

The same figure also suggests that if the start day is within a certain season, then the maximum value is the biggest, in comparison to the other taxa. Thus, it seems that when ALNU, FRAX and BETU, have a starting day between day 70 and day 120, then they reach the biggest pollen count value among the 10 taxa studied. It is interesting to note that the behaviour of some pollen types (like BETU) does not change in terms of peak day and total (accumulative) counts, while for some other taxa, like URTI, the accumulative pollen count is much higher in comparison to the max count recorded (Fig. 1d - 1e). Last, Fig. 1f verifies that there are taxa with a long duration of

existence in the atmosphere (like URTI and POAC), while others (like PLAT) have a much lower duration. All these information are of value for the pollen sufferers, thus meaning that the analysis should continue towards the investigation of the behaviour of the various pollen types.

3.1 Fuzzy Clustering

Over the past years, several methods have been proposed in order to extract fuzzy rules from data such as neuro-fuzzy methods, genetic algorithms for rule selection, and fuzzy clustering using genetic algorithms. The method used for identifying the profile of each pollen type is the fuzzy supervised clustering algorithm, proposed by Abonyi and Szeifert [4]. The method is relative new in the literature, but presents some attractive attributes. The key characteristics of the method are that it uses the class label of each pattern to identify the optimal cluster set and that one rule can represent more than one class with different probabilities, contrary to the common practice in most of the fuzzy rule base systems.

Although this clustering algorithm has been proposed in order to calculate clusters that can be used directly to build a fuzzy classifier, in the context of this research the method is used only to identify the profile of each pollen. Our goal was not to develop a model which can classify data to different types of pollen but to identify the characteristic of each pollen type with respect to several attributes such as the start day, peak day, peak value, etc. To this end, the supervised fuzzy clustering method was used in order to calculate the clusters which describe each pollen type, and from this clusters to extract fuzzy rules that can describe the profile of each pollen type. In the following, we briefly present the basic principles of the algorithm. However, the reader should consider [4] in order to find more details on the algorithm.

The main goal of the algorithm is to partition a set of data into $z_k = [\mathbf{x}_k^T, y_k]$, $k = 1, \dots, N$, with $\mathbf{x}_k = [x_{1,k}, \dots, x_{n,k}]$ and $y_k = [c_1, \dots, c_c]$ into R clusters. The number of available features which describe each data is n and the number of clusters R is chosen by the designer. The fuzzy partition is represented by the partition matrix $U = [\mu_{i,k}]_{R \times N}$ where the $\mu_{i,k}$ element of the matrix represents the degree of the membership, how the z_k belongs to the cluster $i = 1, \dots, R$. The algorithm uses Gaussian membership functions in order to describe the clusters. Each cluster r_i represents a rule of the following form

$$r_i : \begin{array}{l} \text{If } x_1 \text{ is } A_{i,1}(x_{1,k}) \text{ and } \dots \text{ } x_n \text{ is } A_{i,n}(x_{n,k}) \\ \text{then } \hat{y}_k = c_1 \text{ with } P(c_1 | r_i), \dots, \hat{y}_k = c_1 \text{ with } P(c_c | r_i)[w_i]. \end{array} \quad (1)$$

where $P(c_i | r_i)$ is the probability that the r_i th cluster describes the density of the class c_i and w_i is a certainty factor that represents the impact of the rule. Since we do not desire to design a classification model, the certainty factor w_i is omitted from the rest of the analysis.

The clustering is based on the minimization of the sum of weighted $D_{i,k}^2$ squared distances between the data points and the n_i cluster prototypes that contains the parameters of the clusters.

$$\frac{1}{D_{i,k}^2(z_k, r_i)} = P(r_i) \prod_{j=1}^n \exp\left(-\frac{1}{2} \frac{(x_{j,k} - u_{i,j})^2}{\sigma_{i,j}^2}\right) \times P(c_j = y_k | r_i). \quad (2)$$

where $v_i = [u_{1,i}, \dots, u_{n,i}]^T$ denotes the center of the i th cluster, $\sigma_{j,i}^2$ the variance of the Gaussian membership function, and $P(c_j = y_k | r_i)$ the probability that the r_i th cluster describes the density of the class of the k th data.

4 Results and Discussion

Due to numerical problems, only 6 attributes were used, namely start day, peak day, peak value, days, ppp start and ppp end. The centres of the clusters, small green squares, which describe each taxon type in the axes Start day and Days are given in Fig.2.

The extracted rules for each pollen type of the fuzzy supervised clustering method are given in Fig. 3. For each taxon type, six Gaussian membership functions are used in order to describe its characteristic. The Gaussian membership functions have been determined from the fuzzy supervised clustering method in order to model the behaviour of each taxon type to each one of the six attributes.

With the aid of the aforementioned analysis and the membership functions of Fig. 3, it is possible to define fuzzy rules concerning the membership of each one of the studied attributed to the overall data set. Thus for example, if we emphasize on the pollen type, then, for the case of AMBR taxon, we can conclude from Fig. 3 that:

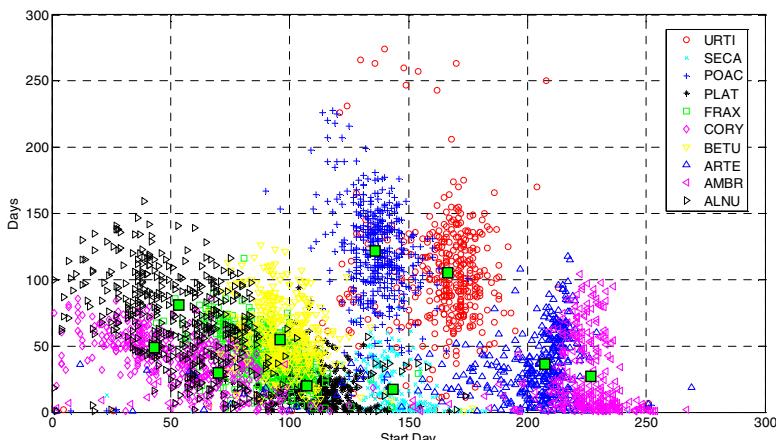


Fig. 2. Pollen type clustering via the fuzzy supervised clustering method

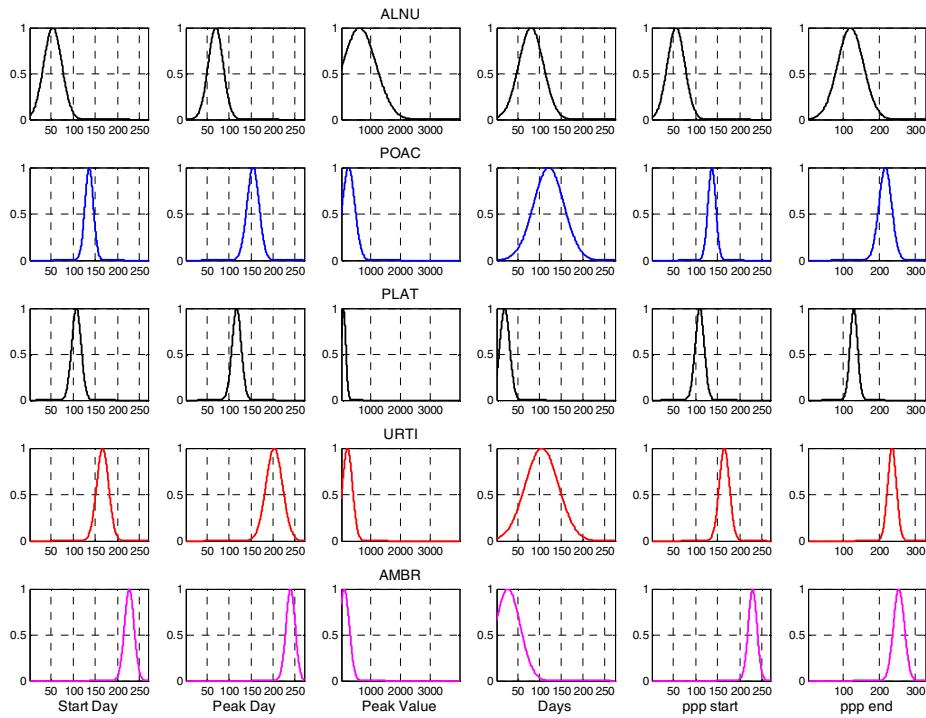


Fig. 3. The corresponding fuzzy Gaussian membership functions for each rule describing the six taxa types

```

IF Start Day is Very Large
AND Peak Day is Very Large
AND Peak Value is Small
AND Days is Small
AND ppp start is Very Large
AND ppp end is Very Large
THEN Particle is AMBR.

```

This means that we can identify the pollen type from its behaviour, even if we have no prior knowledge about its taxon. The confidence level depends from the particle under investigation. For the other pollen types we can derive the corresponding rules using information from similar analysis and figures.

5 Conclusions

It is evident that the application of Computational Intelligence methods, may lead to interesting insights concerning pollen data analysis and knowledge extraction. Within this paper it was possible to show that with the aid of fuzzy algorithms, we can identify the pollen type from its behaviour, even if we have no prior knowledge about its taxon. It should be underlined that the absence of meteorological data did not allow

for the refinement of the analysis and for the development of forecasting models. The collaboration with domain experts (i.e. pollen experts) is expected to lead to much better results, as the focus of the analysis will be fine tuned with the aid of domain expertise.

Acknowledgements

The data used in this paper were based on the work of the following Austrian pollen analysers (in alphabetical order): Bobek Manfred, Bortenschlager Inez, Bortenschlager Sigmar, Brosch Ursula, Cerny Margret, Dutzi Wolfgang, Fritz Adolf, Harvey Pramod, Koll Hertha, Kottik Sabine, Künkele Ute, Langanger Margit, Litschauer Rudolf, Schantl Hanna, Schmidt Jutta, Schmidt Roland, Schultze Ekkehard, and Zwander Helmut.

The authors acknowledge COST Action ES0603 (www.eupollen.eu) for providing them with the opportunity of collaboration.

References

1. Sanchez-Mesa, J.A., Galan, C., Martinez-Heras, J.A., Hervas-Martinez, C.: The use of a neural network to forecast daily grass pollen concentration in a Mediterranean region: the southern part of the Iberian Peninsula. *Clin. Exp. Allergy* 32, 1606–1612 (2002)
2. Ranzi, A., Lauriola, P., Marletto, V., Zinoni, F.: Forecasting airborne pollen concentrations: Development of local models. *Aerobiologia* 19, 39–45 (2003)
3. Aznarte, J.L., Nieto Lugilde, D., Benítez, J.M., Alba Sánchez, F., de Linares Fernández, C.: Forecasting airborne pollen concentration time series with neural and neuro-fuzzy models. *Expert Systems with Applications* 32, 1218–1225 (2007)
4. Abonyi, J., Szeifert, F.: Supervised fuzzy clustering for the identification of fuzzy classifiers. *Pattern Recognition Letters* 24, 2195–2207 (2003)

A New Neural Model for Traffic Simulation

Enrique Mérida-Casermeiro*, Gabriel Aguilera-Venegas,
José Luis Galán-García, and Pedro Rodríguez-Cielos

Department of Applied Mathematics,
University of Málaga, Spain
{merida,gaguilera,jl_galan,prodiguez}@uma.es

Abstract. A new neural model to simulate cars traffic behaviour is introduced in this paper. Several strategies, stated as traffic rules, have been developed in order to improve different important parameters within this matter. This new neural model has been developed with the aim of being close to the natural process. An application for simulating a traffic round with several traffic lanes has been developed using the neural model. Furthermore, it can be easily extended to other similar cases.

Keywords: Traffic simulation, neural model, cellular automata, parallel computing.

1 Introduction

One of the main goals in Engineering is to develop mechanisms that lead to an improvement in the human Quality of Live (QoL). A very important field for QoL is the daily traffic control. Any improvement within this area has high influence in human QoL and it is usually demanded by both users and controllers. Among others we can underline the following problems: delays and traffic jams, slow access to hospital, fire stations, garbage collection, school centres, etc.

Historically, many traffic car models have been designed to describe and take decisions about traffic behaviour in order to improve some related parameters.

Simulating any fixed traffic area is, basically, a continuous stochastic process with a huge number of interrelated time-dependent parameters:

- Number of cars accessing from each possible input to the system.
- Different driving types due to different objectives and behaviours.

Due to their complexity, most of simulating models have been made for large stretches of roads or motorways with one or several lanes, being impossible the use of analytical methods for obtaining solutions, even for these simple problems. In this context, three different types of models have been used:

Macroscopic traffic flow models: Continuous systems of differentials equations are used in order to obtain macroscopic properties for the traffic intensity, the road density, the mean speed, etc. [3]

Microscopic traffic flow models: The traffic behaviour is shown in terms of interacting discrete entities simulating single units. There are two main types:

* Partially supported by the Junta de Andalucía (Spain) under project TIC-01615.

- The car-following models: The behaviour of a car is determined by the leading one. It produces a time-continuous model described by ordinary differential equations. The position and velocity at time $t + \Delta t$ is obtained from the position and velocity of the proper car and the leading one at time t .

- Cellular automaton (CA) models: The space and time are discretised in steps of Δx and Δt . So, the road is divided into sections and, at time t , they can either be occupied by a car or being empty. Different models vary due to changes on the update rules [12].

Kinetic models: The statistic mechanical and kinetic gas theory are the bases of these models. They use the probability of existing a car with a given velocity being allocated in a specific place at time t [45].

Most bibliography approaches have been made by CA microscopic models where they can be highlight the rule 184 [7], the Nagel-Schereckenberg (NS) [1] and the KSSS ones [2]. These methods discretise time and space (generally $\Delta t = 1 s$, $\Delta x = 7.5 m$). Moreover, they consider a discretised velocity v taking values into $\mathcal{V} = \{0, 1, \dots, v_{max}\}$ where v_{max} is the maximum velocity representing the number of space units covered in Δt .

CA models have the ability to reproduce a wide range of traffic phenomena, but the lack of using discrete space and time. But they are simple and numerically efficient, so they are preferred to simulate large road networks in realtime.

1.1 The Nagel-Schereckenberg Model

A section of highroad is modelled as a unidimensional lattice with N cells. At time t , a cell can be free, or busy by a car with velocity $v \in \mathcal{V}$. Updating is made in parallel for all cells considering the value of a cell and its predecessors and considering the following four rules:

- 1) Acceleration: $v_{0t+1} = \min\{v_t + 1, v_{max}\}$. If the maximum velocity has not been reached, any car tends to increase it.
- 2) Braking: $v_{1t+1} = \min\{v_{0t+1}, b\}$, where b is the number of previous free cells (gap). Evidently, the car has not advance more than the gap with the previous car.
- 3) Random braking: With a probability p the velocity is reduced by 1, that is, $v_{t+1} = \max\{v_{1t+1}, 0\}$. If $p = 0$ the model is deterministic, but when $p > 0$, this rule simulates the behaviour of absent-mind drivers.
- 4) Movement: The next position of the car is $x_{t+1} = x_t + v_{t+1}$.

This elemental model simulates an one way highroad, but it can be extended to simulate several ways, traffic accidents, etc. Moreover, the more sophisticated KSSS model can be considered an extension of this one by considering cars placed in several adjacent cells, and the velocity of the leading car in order to adapt the proper velocity to it by anticipating its movement. It uses 3 different random parameters to regulate reaction time in different situations [2].

These models have been used for modelling the traffic waves on roads, motorways and rarely two ways road and crossroads. In this paper we have modelled a generic traffic round inspired by the NS model. The main objective is to observe how the excessive number of cars can collapse it, and phenomena that help to produce that unwanted fact.

In next section the new neural model is described together with the parameters used to monitoring, modelling and managing the traffic round, third section shows some simulation results, and the last section is devoted to conclusions.

2 Modelling the Flow in a Traffic Round

To begin with, space and time of the selected traffic area are discretised. Space is divided into cells of size Δx (the minimum space to contain a car and the security gap when $v = 0$) and time is discretised by Δt (necessary time a driver needs to respond to any incident). So, we obtain:

- The number of lanes C and the number of cells for each lane N_i . We have considered the same N_i for all lanes because it produces minimal perturbations and the model is much more understandable.
- The list L_I of input cells to the traffic round. A car accessing to the round only can get into by specific cells. L will denote its length: $L = \text{length}(L_I)$.
- The list L_O of output cells to the traffic round. A car only can get out the round by some specific cells. The number of these cells is $L' = \text{length}(L_O)$.

So, the traffic round model can be considered as a board with NC squares, or a two-dimensional $N \times C$ array. The proposed neural model must have enough neurons to represent all cars into the round and the different queues (one for each input) of cars pretending access to the round. If the maximum length of any queue is H , a net with $M = NC + HL$ neurons is enough for it.

A neuron is characterised by its state vector $V_i(t) = (N_i, C_i, v_i, I_i, s_i)$ where:

- N_i is an integer number verifying $-L \leq N_i \leq N$, with the following meaning:
 - When $N_i > 0$, the associated car is into the round in position (N_i, C_i) .
 - When $N_i < 0$, the associated car is in the place C_i of queue $|N_i|$.
 - When $N_i = 0$ it indicates the car is not in the system (inactive neuron).
- When $N_i > 0$, $v_i \in \{0, 1, \dots, v_{max}\}$ indicates the velocity and $I_i \in \{-1, 0, 1\}$ points out the turn signal state:
 - $I_i = -1$ indicates the car turn right signal is on. It pretends change to $C_i - 1$ lane, but when $C_i = 1$ it is indicating that its output cell is next.
 - If $I_i = 0$ it is not interested on changing its lane.
 - $I_i = 1$ (turn left signal is on), it is trying to change to $C_i + 1$ lane.
- $s_i \in \{1, 2, \dots, L_O\}$ indicates the desired output (when $N_i \neq 0$).

The neural model state is characterised by the state of all their neurons, so it is represented by a $M \times 5$ matrix. Note that other components can easily be added to V_i to incorporate more interactions, for example, braking lights.

2.1 Computational Dynamics

The 4 first components of the state vector of a neuron are fanned out to all others. With this information calculates in parallel its next state in order to reduce the estimated time to leave the round by the desired output. Next state $V_i(t + 1)$ is differently calculated for active, inactive and queued neurons.

Active neurons: Any active neuron only can go out the round (pass to inactive), remain in the proper cell or advance 1 to v_{max} ones in the same lane or any of two adjacent ones. Next position is expressed by the integer number in $\mathcal{P} = \{1, 2, \dots, N\}$ congruent modulus N .

- If $I_i = 0$, then only cells in the same lane are allowed. The objective for neuron will be to advance the maximum. So, it analyses busy cells in the same lane and it will move to cell $(N_i + v_i(t+1), C_i)$, where d is the minimum distance to other car in the same lane and $v_i(t+1) = \min\{d - 1, v_{max}, v_i(t) + 1\}$.

- When $I_i = 1$ (left turn lights on), then all cells in the upper lane $C_i + 1$ are preferred to the present one. So firstly it analyses busy cells in lane C_{i+1} , but considering preferences for cars in $C_i + 1$ and even, in $C_i + 2$ lane with $I_k = -1$. That is, if exists k with $C_k = C_i + 1$, or $(C_k = C_i + 2) \wedge (I_k = -1)$, then cells $N_k, \dots, N_k + \min\{v_k + 1, v_{max}\}$ will be considered busy. Calling d the distance to the first busy cell on $C_i + 1$ lane, the car will move to $(N_i + v_i(t+1))$ with $v_i(t+1) = \min\{d - 1, v_{max}, v_i(t) + 1\}$, but when $d = 0$, then it will stay in C_i lane, moving in the same form as indicated for $I_i = 0$.

- When $I_i = -1$ (right turn lights on), it must be considered if its output cells is accessible, in that case the next state is $V_i(t+1) = (0, 0, 0, 0, 0)$ (inactive neuron). When its output cells is not accessible, the preferred are those in $C_i - 1$ if $C_i > 1$ lane. So, it must consider the preference for cars into $C_i - 1$ lane, so cells $N_k, \dots, N_k + \min\{v_k + 1, v_{max}\}$ on $C_i - 1$ are busy. If cell $(N_i + 1, C_i - 1)$ is busy, then it moves by the rule given for $I_i = 0$.

Queued neurons: If input cell for k -th input is accessible and there exist queued neurons, neuron with the two first components $(-k, 1)$ (the first neuron in the queue) is activated, and any other in k -th queue reduces the second component $V(t+1) = (-k, C_i(t+1), \dots, s) = (-k, C_i(t) - 1, \dots, s)$.

We have considered the case of both cells $(L_I(k), 1)$ and $(L_I(k) + 1, 1)$ will be free, then next state for the activated neuron is $(L_I(k) + 1, 1, 2, 0, s_i)$, but if only is free $(L_I(k), 1)$, then next state will be $(L_I(k), 1, 1, 0, s_i)$.

Inactive neurons: At any step, a $A_{L \times L'} = (\lambda_{i,j})$ matrix is considered, where $\lambda_{i,j}$ is the Poisson distribution parameter characterising the ratio of cars accessing by i -th input and want to leave by the j -th output. So, an item of a random matrix $A_{L \times L'}$ is obtained and $a_{i,j}$ neurons are placed into i -th queue with j as the fifth component. They are added (randomly sorted) to the bottom of their queues, being allowed to be queued and activated in the same step.

Directional light: The usual situation $I_i = 0$ is modified when the car found an obstacle into its lane that force to reduce velocity drastically, so when $v_i(t+1) - v_i(t) < -2$ or $(v_i(t+1) \leq 1) \wedge v_i(t+1) - v_i(t) \leq 0$, the car wants to change to a more quick lane putting $I_i(t+1) = 1$. On the other hand, when its output is near, it wants to reduce the lane number, or, at least, does not increase it (being this rule preferred to others). In simulations, it puts $I_i(t+1) = -1$ if $D = (L_I(s_i(t+1) - N_i(t+1)) < 20(C_i(t+1) - 1)$, that is, it reserves 20 positions to reduce each lane and when $D < 30(C_i(t+1) - 1)$ it does not increase it.

3 Simulations

We have focused our interest in observing the capacity of the round to absorb traffic flow characterised by the Λ matrix. For that, we have consider an standard round with $N = 100$, $C = 3$, $L_I = \{1, 26, 51, 76\}$, $L_O = \{100, 25, 50, 75\}$ and $v_{max} = 5$. The model is initialised with different traffic density, without queued neurons, but it was observed it was not important, and collapse is mainly generated by the values of the matrix Λ and, with minor importance, by driver's behaviour. The stationary situation is reached in less of 1000 iterations, but 5000 ones have been considered. If collapse is not produced other parameters

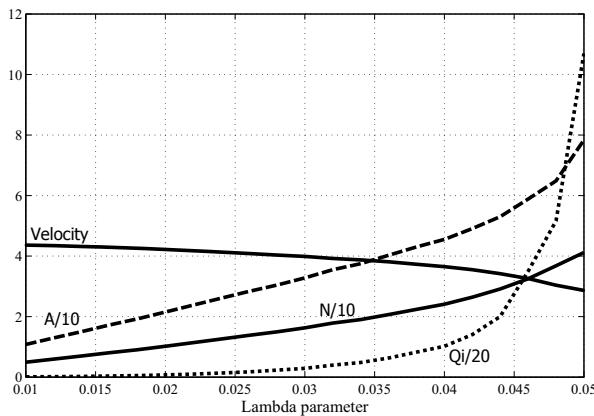


Fig. 1. For example 1 and different λ values, averages of total advanced cells (A), queued in each input (Q_i), velocity and the number of active neurons (N) are represented with different scales

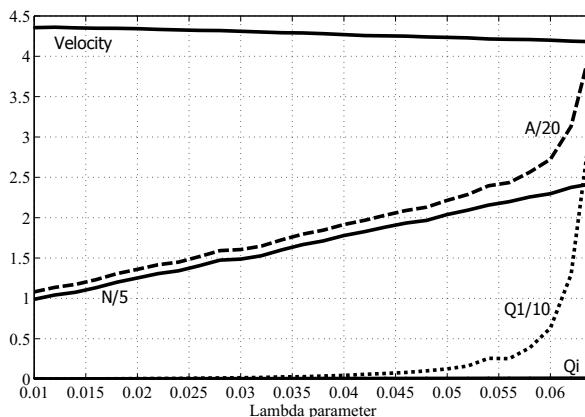


Fig. 2. For example 2 and different λ values, averages of total advanced cells (A), queued in lane 1 (Q_1), queued in other input lanes (Q_i) (values next to 0), velocity and the number of active neurons (N) are represented with different scales

are captured from the model: total number of cars and advances, average velocity, matrix of average time to leave the round,...

Two different essays have been performed (100 times each). In the first one, the matrix $\Lambda_1 = (\lambda_{i,j})$ has $\lambda_{i,i} = \lambda$ and $\lambda_{i,j} = 3\lambda$, with λ varying from 0.01 to 0.05. Increasing the λ parameter implies more cars are accessing to the round and as it can be observed in figure 1, the number of cars into the round, total covered distance and average number of queued cars are also increasing, but average velocity is decreasing. For $\lambda > 0.05$ collapse is always obtained. For the second example the components of matrix $\Lambda_2 = (\lambda'_{i,j})$ are: $\lambda'_{i,i} = 0.01$ and $\lambda'_{i,j} = 0.03$ for $i > 1$, while $\lambda'_{1,1} = \lambda$ and $\lambda'_{1,j} = 3\lambda$ for $j > 1$, varying λ from 0.01 to 0.064. For $\lambda > 0.64$ collapse is always produced due to cars into round do not allow enough access to it by queue 1. Increasing the λ parameter implies more cars are accessing to the round by the first input, however the others stay with low input density. It is shown (see figure 2) how with a minor number of cars into the round, a collapse is produced in queue 1.

4 Summing-Up

The NS model and some improvements to it can be represented by a non standard neural model built specially for each traffic area. Each neuron is associated to a car moving into a board describing the characteristics of the modelled traffic area. Any car/neuron evolves in order to improve its place to leave the area by the desired place. Most characteristics of the state vector constitutes the output of the neuron and then are spanned to any others as inputs, but some others (desired output) are not.

Then some traffic rules and driver's behaviour can be essayed showing the capacity and many average (macroscopic) parameters.

References

1. Nagel, K., Schreckenberg, M.: A cellular automaton model for freeway traffic. *Journal de Physique I* 2, 2221–2229 (1992)
2. Knospe, W., Santen, L., Schadschneider, A., Schreckenberg, M.: Towards a realistic microscopic description of highway traffic. *Journal of Physics A: Mathematical and General* 33(48), 477–485 (2000)
3. Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of modeling and simulation: Integrating discrete event and cont. complex dynamic systems. Academic Press, London (2000)
4. Newell, G.F.: A simplified theory of kinematic waves in highway traffic, part I: General Theory. II Queueing at freeway bottlenecks. *Transportation Research - B* 27(4), 281–287 (1993)
5. Lighthill, M.J., Whitham, G.B.: On kinematic waves. I: Flood movements in long rivers. II: A theory of traffic flow on long crowded roads. *Proceedings Royal Society A* 229, 281–345 (1955)
6. Rodríguez Zamora, R.: Modelación de flujo de tránsito de autos utilizando autómatas celulares. Doctoral Thesis. Puebla-Mexico (2002) (in Spanish)
7. Wolfram, S.: A new class of science. Wolfram Media (2002)

System Architecture for a Smart University Building

Thanos G. Stavropoulos^{1,2}, Ageliki Tsioliardou¹, George Koutitas¹,
Dimitris Vrakas^{1,2}, and Ioannis Vlahavas^{1,2}

¹ School of Science and Technology, International Hellenic University, Thessaloniki, Greece
{atsiolia,g.koutitas}@ihu.edu.gr

² Department of Informatics, Aristotle University of Thessaloniki, Greece
{athstavr,dvrakas,vlahavas}@csd.auth.gr

Abstract. This paper presents a system architecture that provides smart building monitoring and management. The proposed solution integrates heterogeneous geographically disparate sensor networks and devices, and enables optimal operations of the building while reducing its energy footprint. The platform is based on Semantic Web Services composition using AI Planning, that integrates and manages WiFi, RFID and ZigBee networks providing connectivity to the devices. The goal is to develop a model that follows the latest guidelines in the area of Information Communication Technologies (ICT) for sustainable growth, energy efficiency and better quality of life.

Keywords: Sensor Networks, Embedded Systems, Smart Building, Semantic Web Service Composition, Energy Efficiency.

1 Introduction

Information and Communication Technologies (ICT) enable the operation and integration of smart metering devices in complex environments in the form of sensor networks and embedded systems. Applications are met in various domains that target intelligent management, monitoring and improvement of QoL. Characteristic examples are environmental monitoring [1], smart building scenarios [2] and Telemedicine.

In this paper, an intelligent platform is proposed, that integrates sensors within a university building and campus based on Web Services middleware. The aim is to provide automation of common processes, reduce the energy footprint and provide control of devices in a remote manner. The proposed platform, named ‘Smart International Hellenic University’¹, incorporates the ‘Intelligent Building’ concept [2] and the ‘Smart Building’ initiatives that target energy footprint minimization, following guidelines of various FP7 projects, such as Dehems [3] and Hydra [4]. Dehems presents a system architecture for energy efficiency monitoring in different households, mainly focusing on white appliances. Hydra proposes a middleware to expose various devices through Semantic Web Services. Our approach is also based on a Semantic Web Service middleware, which is further enriched with dynamic composition capabilities, proposes specific applications for a university building and facilitates educational processes.

¹ This project is funded by Operational Program Education and Lifelong Learning.

The rest of the paper is organized as follows: In the first section of the paper the ‘Smart University’ concept is presented. This is followed by a review of the available sensor network technologies. Finally, the integration platform is presented with a brief explanation of the system’s components.

2 Smart Building Overview

The Smart Building concept enables remote monitoring and management of processes while providing energy efficiency. The objective of the proposed platform is to design, develop and evaluate a smart building in the International Hellenic University (IHU) and deliver the following services to the end users:

i) Power Consumption Monitoring

An essential step in reducing the energy consumption in a building is the implementation of a measuring and monitoring system. To support such functionality, we intend to monitor the individual electrical devices and appliances of the building, allowing the users to understand/determine further how the energy consumption is distributed among the various IHU facilities. In addition, the system will enable real time monitoring of the University’s data center and provide on line information concerning the Data Center infrastructure Efficiency index (DCiE) and the Data Center energy Productivity (DCeP), as defined by the green grid association and in [5]. Users will be able to convey their information in multiple useful formats, both in past time and in real time, and in different spatial granularity scales, ranging from department-wide, auditorium-wide to appliance-specific consumption characteristics.

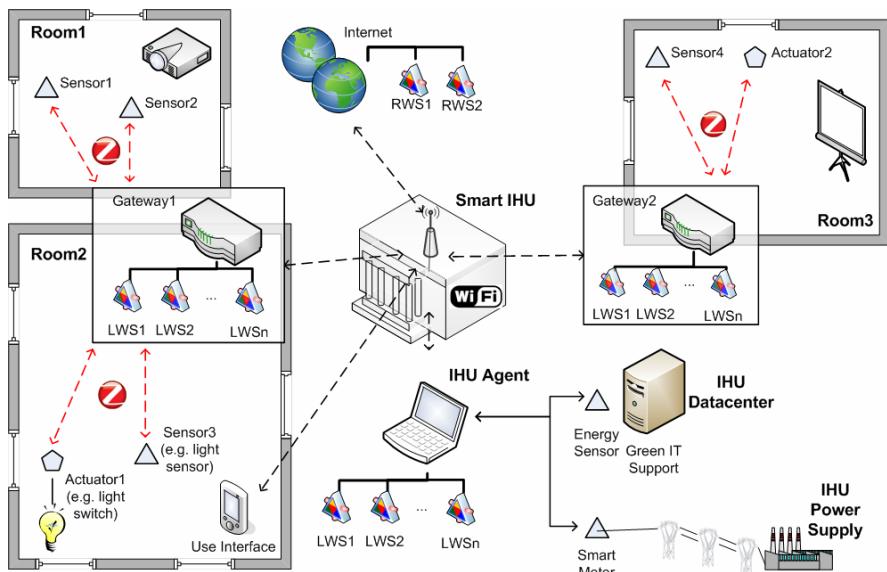


Fig. 1. Architecture of the Smart University System

ii) Energy Efficiency savings

An interactive platform for controlling the energy consumption of nodes is also considered. Through consumption visualization, users are empowered to take energy saving actions based on i) the advanced statistics provided by the system's remote monitoring and optimization capabilities and ii) a simple look at the devices' status.

iii) Building Automation

In the smart building environment, whose architecture is presented in Figure 1, sensors and actuators are deployed spatially in those rooms of the building where monitoring and management is targeted. A number of WiFi/ZigBee gateways serve as both sensors sinks and interoperability agents for the various deployed sensor networks. All building-wide monitoring and controlling capabilities are published in the form of modular web services. The user can access content-based information of the building's parameters based on ZigBee, WiFi and RFID networks. Moreover, a smart metering device is utilized to monitor the building-wide energy consumption.

3 Home Automation Network Technology

The system incorporates different types of sensors that are used to monitor standard environmental attributes, such as humidity, temperature and luminance. These are used to schedule the on/off switching of devices through actuators. Emergencies (e.g. fire, security breaches) are detected by smoke and motion sensors. The aforementioned devices serve as building blocks for the realization of all energy efficiency and automation scenarios.

It is known that the error rates of wireless communication systems can vary widely, depending on the positioning of the devices. Optimal positioning, will be resolved via simulations based on ray-tracing and coverage prediction techniques.

For the purpose of our investigation, the following communication technologies and corresponding protocols were reviewed for possible integration to the present project.

3.1 Power Line Communication Technologies

Power Line Communication (PLC) [6] technology uses the existing power line infrastructures to transmit data and control signals. There are several PLC protocols, aiming at home automation, home security, and lighting control.

X-10 is the oldest common automation protocol over power lines used in homes. It operates by sending 120 kHz signal bursts, each one millisecond, during zero crossings of AC voltage. X-10 is inexpensive and simple to use, however, it only supports a raw data rate of around 50bps and 100 bps in 60 Hz and 50 Hz respectively, while it is very prone to errors.

The Homeplug Alliance has released a series of standards with different Physical Layer modulation techniques. The first broadband power line communication specification HomePlug 1.0 [7] was introduced for home networking, with a rate of 14Mbps. Succeeding specifications are "HomePlug AV" which aim to provide sufficient bandwidth for multimedia applications in residences, "HomePlug BPL" designed for

high-speed Internet access inside residences and “HomePlug Command and Control” that provides a low-speed and low-cost solution for home automation.

The local operation networks (Lonworks) platform [8] was introduced to provide an open interoperability standard among vendors in the control network industry. The physical layer (PHY) signalling can be implemented over a range of communication media, including twisted pair, coaxial cable, fiber, Infrared/Radio Frequency (RF) and power line. The LonWorks PLC technology offers data transmission rate of either 5.4 or 3.6 kbps depending on the frequency.

3.2 Radio Systems

A variety of short-range wireless technologies have emerged, that provide flexible networking patterns suitable for residences. In terms of network control in a smart home, those wireless technologies yield speed, low power consumption, high cost-effectiveness, networking and deployment flexibility, as well as full house coverage.

The Z-Wave technology uses the proprietary 868.42 ISM MHz band in Europe (908MHz in U.S.A) and has a raw data rate of 40Kbps within 100 meters using Binary Frequency Shift Keying (BFSK). It is typically used for consumption monitoring of small houses.

Zigbee [9] is an IEEE 802.15.4-based bidirectional wireless technology for home and building automation domain aiming to reduce energy consumption and prolong wireless sensor battery life [10]. Based on this characteristic, Zigbee technology is preferred for home automation and sensor networking.

Insteon pioneers use both wireless and power line technologies simultaneously with a specially designed protocol. An insteon-based device works on a frequency of 131.65 KHz over the power line for a sustained bit rate of 2.88Kbps and also offers an instantaneous bit rate of 38.4Kbps over the Radio Frequency (RF) 904Mhz band.

Radio Frequency Identification (RFID) tags are used to provide environments that are safer, smarter, and more secure. This technology is the emerging way of controlling information and material flow, especially suitable for large production networks. It relies on the embedment of passive UHF tags in all monitored objects, which enables short distance tracking and low cost network deployments.

Known electrical interference issues of the power line technologies have led to the adoption of radio communication scheme among sensors in the IHU smart building. Furthermore, Zigbee was chosen as the most appropriate related protocol since it offers battery lifetime maximization for the sensors.

4 Information Integration

In the context of various devices, protocols and data formats, there is a need for a common framework that addresses interoperability issues. Most ambient intelligence approaches are based on the Service Oriented Architecture (SOA), where each module of the system is wrapped by a service defining the interface of the module in terms of protocols and functionality.

Nowadays, Web Services [12] are the preferred realization of the basic notion of SOA. As platform-independent APIs, they can be used to facilitate universal remote

access to the heterogeneous sensor data as well as control of the system's actuators. Indeed, there are numerous ambient intelligence approaches implementing Web Services as a middleware that serves as a bridge between the various devices and a centralized computing module embodying the intelligence of the system [11, 13, 14].

Web Service descriptions can be vague or misleading, thwarting their efficient use by human users and especially by machines. Therefore, it is important to enhance the definitions of their functionality and interface with well formed metadata. The notion of Semantic Web Services [15] emerged through the evolution of the Semantic Web and Ontologies. Apart from the syntactic meaning of the exchanged messages, they also define the semantic meaning and exploit ontologies while their descriptions come in various languages, like OWL-S¹ and WSDL-S².

The proposed architecture, as presented in Figure 1, facilitates the integration of heterogeneous data by associating each sensor and actuator with Semantic Web Services, described in OWL-S and based on an ontology manager hosted in the IHU Agent. The smart IHU is based on five categories of services:

- a. Local information services (LIS) that connect to the sensors and provide their measurements.
- b. Local Action Services (LAS) that connect to the actuators and realize the requested adjustments (e.g. value on a thermostat)
- c. Local Computation Services (LCS) that perform complex computational tasks (e.g. face recognition on the data from a camera)
- d. Global Computation Services (GCS) that also perform computational tasks but are hosted outside the campus (e.g. public safety services)
- e. Global Providers Services (GPS) that are hosted by various providers (e.g. local gas station) and allow the automatic placement of orders.

LIS and LAS are initially placed in the corresponding gateways, since there is currently limited or no availability of sensors and actuators with computing and TCP/IP capabilities. LCS are hosted by the IHU Agent or other computers connected to the local TCP/IP network, and GCS and GPS are hosted in computers located outside the building.

The IHU Agent is responsible for carrying out complex tasks that facilitate everyday activities of the students and personnel of the IHU. In order to automate these tasks, the agent employs service composition techniques using AI Planning, which has proven to be a very promising approach [16, 17].

5 Conclusion

This paper presented a framework for developing a Smart University application, incorporating an energy-efficient, sensor and actuator network, and an appropriate Information Integration methodology, based on Semantic Web Service Composition. It enables real-time monitoring and control of devices towards energy-awareness and optimization. Future research essentially includes implementation of the proposed

¹ <http://www.w3.org/Submission/WSDL-S/>

² <http://www.w3.org/Submission/OWL-S/>

platform and evaluation of the equipment, network topology and ontology design, in order to exploit the system's Services, in the most efficient, dynamic and fault-tolerant way.

References

1. Karatzas, K., Moussiopoulos, N.: Development and Use of Integrated Air Quality Management Tools in Urban Areas with the aid of Environmental Telematics. *Environmental Monitoring and Assessment* 65, 451–458 (2000)
2. Chatzigiannakis, I., Koninis, C., Mylonas, G., Colesanti, U., Vitaletti, A.: A Peer-to-Peer Framework for Globally-Available Sensor Networks and its Application in Building Management. In: Proc. of the 2nd International Workshop on Sensor Network Engineering (IWSNE 2009), Marina Del Rey, CA, USA (2009)
3. The Dehems project, <http://www.dehems.eu>
4. The Hydra project, <http://www.hydramiddleware.eu>
5. Koutitas, G., Demestichas, P.: Challenges for energy efficiency in local and regional data centers. *Journal on Green Engineering* (to appear, 2010)
6. Yousuf, M.S., El-Shafei, M.: Power Line Communications: An Overview - Part I. In: Proc. of the 4th Int. Conference on Innovations in Information Technology, pp. 218–222 (2007)
7. Lee, M.K., Newman, R., Latchman, H.A., Katar, S., Yonge, L.: HomePlug 1.0 Powerline Communication LANs –Protocol Description and Comparative Performance Results. *International Journal on Communication Systems* 6(5), 447–473 (2003)
8. Amitava, D.-R.: Networks for home. *IEEE Spectrum* 36(12), 26–33 (1999)
9. Baronti, P., Pillai, P., Chook, V.W.C., Chessa, S., Gotta, A., Hu, Y.F.: Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communication* 30(7), 1655–1695 (2007)
10. Lee, J.S.: Performance evaluation of IEEE 802.15.4 for low-rate wireless personal area networks. *IEEE Trans. Consumer Electron* 52(3), 742–749 (2006)
11. Urbieta, A., Barrutieta, G., Parra, J., Uribarren, A.: A survey of dynamic service composition approaches for ambient systems. In: Proc. of the Ambi-Sys Workshop on Software Organisation and MoniToring of Ambient Systems, pp. 1–8 (2008)
12. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., et al.: Web Service Architecture, W3C Working Group Note 2004 (2004), <http://www.w3.org/TR/ws-arch/>
13. V. Issarny , Sacchetti , Tartanoglu , Sailhan , Chibout , Levy , Talamona,
14. Bellur, Nanjangud, N.: Towards Service Orientation in Pervasive Computing Systems, Umesh. In: Proc. of the International Conference on Information Technology (ITCC 2005), Las Vegas, USA, pp. 289–295 (April 2005)
15. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic Web Services. *IEEE Intelligent Systems* 16(2), 46–53 (2001)
16. Lecue, F., Leger, A.: Semantic Web Service Composition Based on a Closed World Assumption. In: Proc. of the European Conference on Web Services, pp. 171–180 (2006)
17. Hatzis, O., Meditskos, G., Vrakas, D., Basiliades, N., Anagnostopoulos, D., Vlahavas, I.: Porsche II: Using Planning for Semantic Web Service Composition. In: Proc. of the ICCEPS2009 in Conjunction with ICAPS 2009, Thessaloniki, Greece (2009)

Monitoring and Assessment of Environmental Impact by Persistent Organic Pollutants

Jaroslav Urbánek¹, Karel Brabec¹, Ladislav Dušek², Ivan Holoubek¹, Jiří Hřebíček², and Miroslav Kubásek²

¹ Masaryk University, Research Centre for Toxic Compounds in the Environment,
Kamenice 126/3, 625 00 Brno, Czech Republic

{urbanek, brabec, holoubek}@recetox.muni.cz

² Masaryk University, Institute of Biostatistics and Analyses,

Kamenice 126/3, 625 00 Brno, Czech Republic

{dusek, hrebicek, kubasek}@iba.muni.cz

Abstract. Institute RECETOX coordinates the “Central and Eastern European Centre for Persistent Organic Pollutants (POPs)” of Stockholm Convention (SC) environmental services: data monitoring, processing, storage and their management. This is supported by the Global Environmental Assessment and Information System (GENASIS) that utilizes data from national and international monitoring networks to obtain as-complete-as-possible set of information and a representative picture of environmental contamination by POPs. Institute of Biostatistics and Analyses (IBA) operates the System for Visualizing of Oncology Data (SVOD) for cancer epidemiology in the Czech Republic for data from Czech National Oncology Register. The synthesis of existing POPs pollution monitoring databases with epidemiological data is required for identifying some effects of POPs on human health. This task requires new, rich, data and services discovery capabilities within the bodies of knowledge available, which are discussed in the paper. IBA and RECETOX anthropogenic impact studies requiring data discovery from a multitude of monitoring networks and resources. The FP7 project “TaToo - Tagging Tool based on a Semantic Discovery Framework” are setting up a semantic web solution to close the discovery gap that prevents a full and easy access to information resources on the web. The use of TaToo tools together with GENASIS and SVOD is presented for the discovery of anthropogenic impact in several examples.

Keywords: GENASIS, SVOD, TaToo, POPs, Stockholm Convention, human health issues, monitoring, MONET.

1 Introduction

Persistent organic pollutants (POPs) represent a long-term problem which is connected with the production, application, and disposal of many hazardous chemicals. POPs have several undesirable properties. They persist in the environment for a long time (e.g. several years, or even decades of years), they bioaccumulate, they are toxic, and they are subject to long-range transport. *Research Centre for Toxic Compounds in*

the Environment (RECETOX) is focused on the research of the fate and biological effects of these and other toxic substances in the environment. RECETOX monitors these chemicals in several matrices including air, soil, water, or human milk. RECETOX supports (among others) implementation of international conventions on chemical substances like the *Stockholm Convention on Persistent Organic Pollutants* (POPs, <http://chm.pops.int/>). It is also a regional centre of the Stockholm convention for Central and Eastern Europe.

In January 2010 RECETOX launched the first version of the *Global Environmental Assessment and Information System* (GENASIS, <http://www.genasis.cz>) [1], which provides information support for implementation of the Stockholm convention at international level. Initial phase of the GENASIS project is focused on data from regular monitoring programmes, providing a general overview of spatial patterns and temporal trends of pollutants concentrations.

RECETOX closely cooperates with the *Institute of Biostatistics and Analyses* (IBA). IBA is a research institute focused on the solution of scientific projects and providing related services especially in the field of biological and clinical data analysis organization and management of clinical trials, software development and ICT applications. IBA created a web portal of epidemiology of malignant tumours in the Czech Republic, SVOD (System for Visualizing of Oncological Data, <http://www.svod.cz/>) [4], based on the data from the Czech National Oncology Register.

We know that POPs are harmful and some of them can cause a cancer. The aim is now to try to find out whether there is a connection between the concentration of POPs and cancer occurrence in some regions. This task requires new discovery facilities which will be developed within the FP7 project called TaToo (Tagging Tool based on a Semantic Discovery Framework).

2 GENASIS

GENASIS currently (in the version 1.0) contains information about the aims of this project and about Stockholm convention, encyclopaedic description of the substances included in the Stockholm convention, analytical module, related case studies and scientific topics. The system is freely accessible to anyone with all its functionality.

2.1 Analytical Tools

One of the main parts of the GENASIS system is an interactive analytical module. Its tools allow viewing of data and performing several statistical analyses with selected data. The module is based on GENASIS database which contains (in the version 1.0) data from two projects of POPs monitoring in ambient air. The first project, Košetice, representing both active and passive air sampling with the data containing chemical concentrations in the gas phase as well as in the particulate phase is a unique continuous monitoring of POPs since 1995. The second project, called MONET [6], represents a monitoring network of POPs in ambient air in the Central and Eastern Europe, Africa and Pacific Islands using the passive air sampling technique. The MONET network currently covers 65 countries with more than 350 sampling sites.



Fig. 1. Analysis of temporal trends (left) and seasonal pattern analysis (right) in GENASIS

The initial window of the analytical GENASIS module is a selection window where several items such as matrix, project, region, or time scale have to be selected. After this selection is done, a user can proceed to data analysis (or only to a visualisation of the current selection). The user can choose from several tools to analyse the selected data. There are for example data overview tools for visualisation of localities geographical position, definition of a subset from all sites, and their basic statistical characterization.

Other tools allow the user to view distribution characteristics, time series, and to analyse seasonal and temporal trends. It is also possible to make an additional selection to analyse the sites belonging to one specific group only (e.g. the sites which have the same concrete land use etc.), or to stratify the selection according the same criteria (which are available for an additional selection). All the analytical tools (where this makes sense) generate a basic descriptive statistics table. The user can also choose from several analysis settings such as the chart settings or setting the level of aggregation (month, quarter, or year). Two examples of data analysis are shown in Figure 1.

2.2 Data Model

GENASIS database can be used for all environmental data although it is currently primary linked to persistent organic pollutants. Each record has to fulfil so called

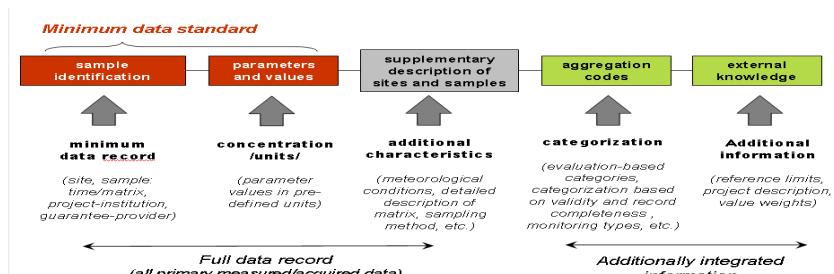


Fig. 2. Record attributes and defined types of data records in Genasis database

minimum data standard to be relevant. This minimum data standard contains: the name of site (where the sample was taken), time, matrix, project/institution, guarantee/provider, substance, and measured concentration. Each data record can have (and in GENASIS always has) additional attributes extending the information related to the sample. Figure 2 shows record attributes and defined types of data records graphically.

3 SVOD

The SVOD portal primary aims to provide representative data about the incidence of malignant tumors in the Czech Republic and about the mortality which is connected to the malignant tumors. It is freely accessible (like GENASIS, but it has also a secure part accessible only after logging in). Its analyses are based on *National Oncologic Register* (NOR) which is managed by Institute of Health Information and Statistics (UZIS CR) of the Czech Republic. It provides validated epidemiological data from the years 1977 - 2007. This represents a unique data set in Europe at least (currently there is more than 1.5 mil. records). Epidemiological trends cannot be created without relevant demographic data about examined population. This data is acquired from the Czech Statistical Office (CSU) on the basis of general agreement about cooperation with Masaryk Memorial Cancer Institute in Brno and Masaryk University Brno.

3.1 Analytical Tools

The main part of the SVOD portal offers a set of analytical tools with graphical and tabular outputs. User can easily analyse epidemiological trends examined over three decades, stratify and filter cohorts of patients and extract population risk in absolute or age-specific values. Some of analyses offer benchmarking with the respect to clinical status of the disease. Major epidemiological trends are available in comparison with international data (GLOBOCAN 2002: Cancer Incidence, Mortality and Prevalence Worldwide) [3].

4 TaToo

TaToo, which stands for “*Tagging Tool based on a Semantic Discovery Framework*” [2], is an FP7 project sharing the vision of a *Single Information Space in Europe for the Environment* (SISE) [5]. It aims to develop tools allowing third parties to easily discover environmental resources on the web and to add valuable information on to these resources. TaToo tools will be validated in three different real-world applications. Validation scenario called “*Anthropogenic impact and global climate change*” belonging to Masaryk University is one of them. This scenario aims to improve the discovery of scientific resources and tries to find relationships between different domains, environmental pollution and epidemiology.

4.1 Use of TaToo Tools in the Anthropogenic Impact Assessment

To evaluate the impact of chemicals on human health, we need to have relevant information corresponding in time and space. The ideal scenario of how TaToo tool

would work is to enable users to search for chemicals and their concentrations in the area and time of interest (the incidence of human illnesses respectively) and provide them with information about possible connection between chemicals concentrations and human illnesses where it can be the case. This connection can serve for further expert investigation if the chemical is really responsible for the health issues in the selected area.

4.2 Use of TaToo Together with GENASIS and SVOD

As a practical example of how TaToo tool should work we can take GENASIS and SVOD together. TaToo users may be for example interested in the chemical called benzo(a)pyrene and its concentrations in the environment of the region where they live and for example for all times available. TaToo tool will return the metadata of found resources. Let us imagine the GENASIS resource is the only one. The metadata will report to users about this resource, its quality, availability of data and services, etc. Users can browse this resource to obtain information they were searching for. Then, they will have the possibility to establish a “find connections to human illnesses” search. It is known that benzo(a)pyrene can cause a lung cancer. The TaToo tool will therefore search in SVOD portal how the incidences of C34 diagnosis (Malignant neoplasm of bronchus and lung) in time in the same region look like. When there are similar trends, then TaToo tool will provide users with this information containing both resources together and a note that due to mentioned reasons there might be a connection. Very similar situation will happen when users are interested in incidences of some illness (for example concrete cancer diagnosis) in a region over the given time. TaToo tool will find a SVOD resource which users can investigate. Then they will have the possibility to find the connection to the potential source of these incidences in terms of chemical concentrations. In this case the TaToo tool will search the GENASIS web site for similar trends in concentrations of chemicals which can cause a given illness. As already mentioned, the output of “TaToo connection search” then needs to undergo further expert investigation. If the results are proven, then appropriate measures have to be taken to remedy the situation. In any case, the TaToo tool would be a very useful instrument for decision support of administrative officers and secondarily for improving the quality of life. Furthermore, the use of TaToo tools can also lead to the discovery of new linkages.

5 Conclusions

The synthesis of existing POPs pollution monitoring databases of GENASIS system with epidemiological data of SVOD system is introduced for identifying some effects of POPs on human health (anthropogenic impacts). This task requires new, rich, data and services discovery capabilities within the bodies of knowledge available, which are also discussed in the paper. Research institutes IBA and RECETOX study anthropogenic impact using data discovery from a multitude of monitoring networks and resources. The FP7 project “TaToo – Tagging Tool based on a Semantic Discovery Framework” which aims to set up a semantic web solution to close the discovery gap that prevents a full and easy access to information resources on the web was introduced. The use of

developed TaToo tools together with GENASIS and SVOD is presented for the discovery of anthropogenic impact in several examples.

Acknowledgments. This work was supported by CETOCOEN (CZ.1.05/2.1.00/01.0001) project granted by the European Union and administered by the Ministry of Education, Youth and Sports of the Czech Republic (MoEYS), the project INCHEM-BIOL (MSM0021622412) granted by MoEYS, and the project FP7 No. 247893 TaToo – Tagging Tool based on a Semantic Discovery Framework) granted by European Commission.

References

1. Brabec, K., Jarkovský, J., Dušek, L., Kubásek, M., Hřebíček, J., Holoubek, I., Čupr, P., Klánová, J.: GENASIS: System for the Assessment of Environmental Contamination by Persistent Organic Pollutants. In: EnviroInfo 2009. Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools. 23. International Conference on Informatics for Environmental Protection, pp. 369–376. Shaker Verlag, Aachen (2009)
2. Brabec, K., Hřebíček, J., Schimak, G., Urbánek, J.: TaToo project: Tagging Tool on a Semantic Discovery Framework. In: 6. Summer School of Applied Informatics, pp. 6–16. Littera, Brno (2009)
3. Dušek, L., Mužík, J., Kubásek, M., Koptíková, J., Brabec, P., Žaloudík, J., Vyzula, R.: The national web portal for cancer epidemiology in the Czech Republic. In: Proceedings of the 19th International Conference Informatics for Environmental Protection (Enviroinfo 2005). Masaryk University, Brno (2005)
4. Dušek, L., Mužík, J., Kubásek, M., Koptíková, J., Žaloudík, J., Vyzula, R.: Epidemiology of malignant tumours in the Czech Republic [online]. Masaryk University, [2005], On-line available: <http://www.svod.cz>, [cit. 2008-12-15], Version 7.0 [2007], ISSN 1802 - 8861
5. Hřebíček, J., Pillmann, W.: Shared Environmental Information System and Single Information Space in Europe for the Environment: Antipodes or Associates? In: Proceedings of the European Conference of the Czech Presidency of the Council of the EU TOWARDS eENVIRONMENT. Opportunities of SEIS and SISE: Integrating Environmental Knowledge in Europe, pp. 447–458. Masaryk University, Brno (2009)
6. Klánová, J., Čupr, P., Holoubek, I., Borůvková, J., Přibylová, P., Kareš, R., Kohoutek, J., Dvorská, A., Komprda, J.: Towards the Global Monitoring of POPs - Contribution of the MONET Networks. RECETOX, Masaryk University, Brno, Czech Republic (2009), ISBN 978-80-210-4853-9

A Feature Selection Method for Air Quality Forecasting

Luca Mesin, Fiammetta Orione, Riccardo Taormina, and Eros Pasero

¹ Department of Electronics, Politecnico di Torino, Torino, Italy
{luca.mesin,fiammetta.orione,riccardo.taormina,
eros.pasero}@polito.it

Abstract. Local air quality forecasting can be made on the basis of meteorological and air pollution time series. Such data contain redundant information. Partial mutual information criterion is used to select the regressors which carry the maximal non redundant information to be used to build a prediction model. An application is shown regarding the forecast of PM₁₀ concentration with one day of advance, based on the selected features feeding an artificial neural network.

Keywords: Air pollution, artificial neural network, partial mutual information, information theory, input variable selection.

1 Introduction

European laws require the analysis and implementation of automatic procedures to prevent the risk for the principal air pollutants to be above alarm thresholds in urban and suburban areas (e.g. the Directive 2002/3/EC for ozone or the Directive 99/30/CE for the particulate matter with an aerodynamic diameter of up to 10 µm called PM₁₀).

Two-three days forecasts of critical air pollution conditions would allow an efficient choice of countermeasures to safeguard citizens' health.

Different procedures have been developed to forecast the time evolution of air pollutant concentration. Meteorological data are usually included in the model as air pollution is highly correlated with them (Cogliani, 2001). Indeed, pollutants are usually entrapped into the planetary boundary layer (PBL), which is the lowest part of the atmosphere. It is directly influenced from soil interaction, as friction and energetic exchange.

Important meteorological parameters involved in air pollution dynamics are air temperature, relative humidity, wind velocity and direction, atmospheric pressure, solar radiation and rain. Principal air pollutants whose concentration should be monitored are Sulphur Dioxide SO₂, Nitrogen Dioxide NO₂, Nitrogen Oxides NO_x, Carbon Monoxide CO, Ozone O₃ and Particulate Matter PM₁₀. Local forecasting can be performed in real time and with low cost technology analyzing time series of weather data and air pollution concentrations. Meteorological and air pollution data contain redundant information. The introduction of irrelevant and redundant information is detrimental for a prediction model, as the training and processing time are increased and noise is introduced, so that accuracy is reduced (May et al., 2008). Hence, a careful selection of useful predictors is needed.

In this paper, we are concerned with a specific method to perform local prediction of air pollution. Our analysis carries on the work already developed by the NeMeFo (Neural Meteo Forecasting) research project for meteorological data short-term forecasting (Pasero et al., 2004). Special attention is devoted to the selection of optimal, non redundant features (Guyon and Elisseeff, 2003) from meteorological and air pollution data, decisive to describe the evolution of the system.

Our approach for feature selection is based on the partial mutual information (PMI) criterion (Sharma 2000). Once selected decisive features, prediction is performed using an Artificial Neural Network (ANN). ANNs have been often used as a prognostic tool for air pollution (Perez et al., 2000; Božnar et al., 2004; Cecchetti et al., 2004; Slini et al., 2006; Karatzas et al., 2008). The method is applied on four hours hourly data, measured by a station located in the urban area of Goteborg, Sweden (Goteborgs Stad Miljo). The aim of the analysis is forecasting the average day concentration of PM₁₀.

2 Methods

2.1 Input Variable Selection with Partial Mutual Information

2.1.1 Mutual Information

The input variable selection algorithm relies on Partial Mutual Information criterion first introduced by Sharma (2000). The proposed algorithm evolves the concept of Mutual Information (Cover and Thomas, 1991) between two random variables, suitable measure of dependence between signals in nonlinear systems. The mutual information MI(X;Y) of two random variables can be defined as the reduction in uncertainty with respect to Y due to observation of X. For continuous random variables, the MI score is defined as:

$$MI(X;Y) = H(X) + H(Y) - H(X,Y) \quad (1)$$

where H is the information entropy defined as:

$$H(Z) = -\int f_Z(z) \ln f_Z(z) dz \quad (2)$$

Mutual Information is always nonnegative and it equals zero only when X and Y are independent variables.

For any given bivariate random sample, the discrete estimation of (1) can be written as:

$$MI(X;Y) = \frac{1}{n} \sum_{i=1}^n \ln \frac{f_{X,Y}(x_i, y_i)}{f_X(x_i)f_Y(y_i)} \quad (3)$$

where x_i and y_i are the i^{th} bivariate sample data pair in a sample of size n , and $f_X(x_i)$, $f_Y(y_i)$ and $f_{X,Y}(x_i, y_i)$ are respective univariate and joint probability-densities estimated at the sample data points. Robust estimators for the joint and marginal probability density functions in (3) can be computed with Parzen method (Parzen, 1962; Costa et al., 2003):

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i) \quad (4)$$

where d is the number of dimension of the considered random variable and the kernel K was assumed as Gaussian with standard deviation h (kernel bandwidth). A major issue in kernel density estimation is to find the value for the bandwidth. Small values could lead to data under-smoothing and the resulting MI score could be noise-sensitive thus. A large bandwidth tends to over-smooth the sample probability density-function and underestimate MI value consequently. Although algorithms have been developed to search for an optimal value of the bandwidth h , an appropriate first choice is given by the Gaussian reference bandwidth:

$$h = \left(\frac{4}{d+2} \right)^{\frac{1}{d+4}} \sigma n^{-\frac{1}{d+4}} \quad (5)$$

where σ is the standard deviation of the data sample.

2.1.2 Partial Mutual Information

Optimal input set for modeling a certain system can be defined selecting the variables with large Mutual Information with the output. However, this raises a major redundancy issue because the MI criterion does not account for the interdependencies between candidate variables. To overcome this problem, Sharma (2000) has developed an algorithm that exploits the concept of Partial Mutual Information (PMI), which is the nonlinear statistical analog of partial correlation. It represents the information between two observations that is not contained in a third one.

Let Y be an univariate random variable with observations y , X a candidate input with observations x , and Z the multivariate set of the input variables which have already been selected as predictors. The PMI score between candidate X and output Y is computed considering the residuals of both variables once the effects of the existing predictors Z have been taken into account. In other words the arbitrary dependence between variables is removed by computing for each x and y the residuals:

$$x' = x - E[x|z] \quad y' = y - E[y|z] \quad (6)$$

where $E[.]$ denotes the regression of the chosen variable based on the predictors z already selected (i.e., belonging to Z). Using the kernel density estimation approach, the output Y can be estimated as:

$$E[y|z] = \frac{1}{n} \frac{\sum_{i=1}^n y_i K_h(z - z_i)}{\sum_{i=1}^n K_h(z - z_i)} \quad (7)$$

The regression estimator $E[x|z]$ for the candidate predictors x to be possibly included in Z is written analogously.

2.1.3 Termination Criterion

The above mentioned approach needs a criterion to assess whether each selected variable is indeed a significant predictor for the system output. Different methods to

terminate the algorithm have been proposed, e.g. bootstrap estimation technique (Sharma, 2000) or less computationally intensive approaches (May et al., 2008). This work applied the Hampel test criterion, which is a modification of the Z-test commonly adopted to find outliers within a population of observed values (May et al., 2008).

2.2 Prediction Method Based on Artificial Neural Network

The prediction algorithm was based on feedforward ANNs with a single hidden layer and a single output (the predicted concentration of pollutant). The hyperbolic tangent function was used as activation function. The Levenberg-Marquardt algorithm (Haykin, 1999) was used to estimate iteratively (using backpropagation) the synaptic weights of the ANN, minimising the sum of the squares of the deviation between the predicted and the target values on a training set.

For prediction purposes, time is introduced in the structure of the neural network. For just further prediction, the desired output at time step n is a correct prediction of the value attained by the time series at time n+1:

$$y_{n+1} = \varphi(\vec{w} \cdot \vec{z}_n + b) \quad (8)$$

where the vector of regressors \vec{z}_n includes information available up to the time step n. Selected features up to time step n were used, obtaining a non linear autoregressive with exogenous inputs (NARX) model (Sjöberg et al., 1994).

3 Results

The input selection method was tested on a dataset for the prediction of the daily average PM₁₀ in Goteborg, urban area. Apart from values of PM₁₀ itself, other candidate variables were both meteorological (air temperature, relative humidity, atmospheric pressure, solar radiation, rainfall, wind speed and direction) and chemical ones (SO₂, NO_x, NO₂, O₃ and CO). Daily averages, three previous days maximum and minimum have been included for each variable and 24-hour cumulated variable only for the rain. In this way, the candidate pool was made of more than 120 features, with observations ranging from the beginning of 2002 to the end of 2005. First three years of the database have been used for selecting best input variables, while last year recordings have been arranged for testing the performances of the ANN developed for prediction. The results of the PMI algorithm on the candidate dataset are reported in Fig. 1A. Only three variables were selected using Hampel test, namely the maximum, daily average and minimum concentration of PM₁₀ recorded the day before. This entails a drastic reduction from the original pool of candidates.

Once the most significant features have been selected, an ANN has been developed to predict future values of the PM₁₀ concentration, using the same dataset of the input selection algorithm for the training. The optimal ANN was found to have 8 hidden neurons, and the results on the test dataset are plotted in Fig. 1B. Root mean square error and correlation coefficient on the test data set were respectively RMSE=6.24 $\mu\text{g}/\text{m}^3$ and CC = 0.91, showing an overall good fitting of the ANN output.

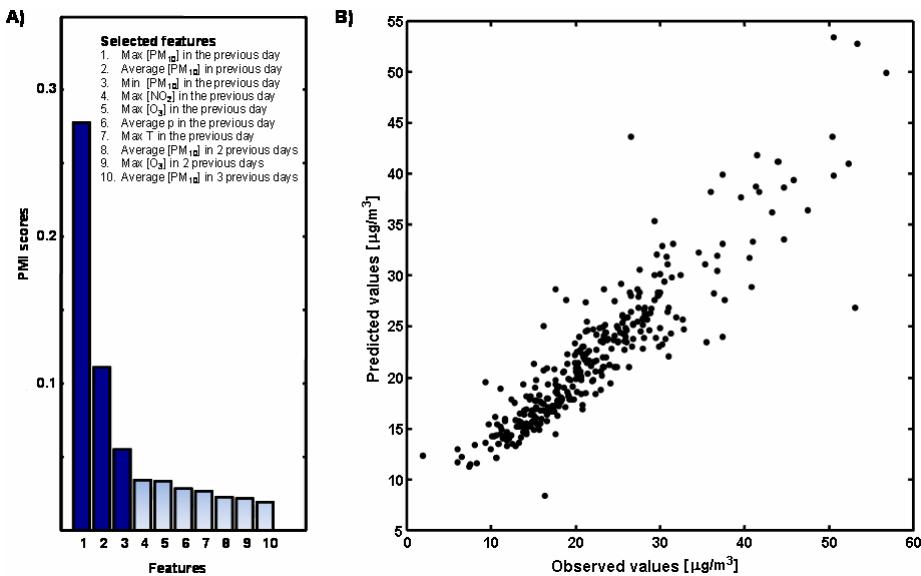


Fig. 1. A) Partial mutual information scores for the 10 most significant features. The first three features were selected by the Hampel test. B) Comparison between the observed and predicted PM_{10} concentration.

4 Discussion

Coarse fraction of PM_{10} derives most from natural source, as wind erosion, sea salt spray, wood waste. Finest fraction of PM_{10} derives most from human activities as transport action or construction ones. Nitrates and sulphur dioxides are found predominantly in fine fraction, less than $2.5 \mu\text{m}$ in diameter.

Our model selected previous day maximum, minimum and average concentration of PM_{10} as the most important features of which taking account in PM_{10} daily monitoring. This suggests that most of the information needed to forecast future values is indeed contained in the trends of the pollutant itself. In addition, two or more days before observations do not seem to have explaining potential for the prediction. Nitrates are found to have a relative high PMI, although the score is below the threshold for being selected. Meteorological variables have not been selected as well. Due to good prediction performance, this may imply that the meteorological effects as well as chemical interactions between the pollutants might be included in the information provided by the PM_{10} features.

Acknowledgments. This work was sponsored by the national project AWIS (Airport Winter Information System), funded by Piedmont Authority, Italy.

References

- Božnar, M.Z., Mlakar, P.J., Grašič, B.: Neural Networks Based Ozone Forecasting. In: Proceeding of 9th Int. Conf. on Harmonisation within Atmospheric Dispersion Modelling for Regulatory Purposes, Garmisch-Partenkirchen, Germany, June 1-4 (2004)

2. Cecchetti, M., Corani, G., Guariso, G.: Artificial Neural Networks Prediction of PM10 in the Milan Area. In: Proc. of IEMSS 2004, University of Osnabrück, Germany, June 14-17 (2004)
3. Cogliani, E.: Air pollution forecast in cities by an air pollution index highly correlated with meteorological variables. *Atm. Env.* 35(16), 2871–2877 (2001)
4. Costa, M., Monaci, W., Pasero, E.: INFO: an artificial neural system to forecast ice formation on the road. In: Proceedings of IEEE International Symposium on Computational Intelligence for Measurement Systems and Applications, July 29-31, pp. 216–221 (2003)
5. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons, New York (1991)
6. Goteborgs Stad Miljo, <http://www.miljo.goteborg.se/luftnet/>
7. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *The Journal of Machine Learning Research* 3, 1157–1182 (2003)
8. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, Englewood Cliffs (1999)
9. Hyvärinen, A.: Survey on Independent Component Analysis. *Neural Computing Surveys* 2, 94–128 (1999)
10. Karatzas, K.D., Papadourakis, G., Kyriakidis, I.: Understanding and forecasting atmospheric quality parameters with the aid of ANNs. In: Proceedings of the IJCNN, Hong Kong, China, June 1-6, pp. 2580–2587 (2008)
11. May, R.J., Maier, H.R., Dandy, G.C., Gayani Fernando, T.M.K.: Non-linear variable selection for artificial neural networks using partial mutual information. *Envir. Mod. And Soft.* 23, 1312–1326 (2008)
12. Parzen, E.: On Estimation of a Probability Density Function and Mode. *Annals of Math. Statistics* 33, 1065–1076 (1962)
13. Pasero, E., Monaci, W., Meindl, T., Montuori, A.: NeMeFo: Neural Meteorological Forecast. In: Proceedings of SIRWEC 2004, 12th International Road Weather Conference, Bingen (2004)
14. Perez, P., Trier, A., Reyes, J.: Prediction of PM2.5 concentrations several hours in advance using neural networks in Santiago, Chile. *Atmospheric Environment* 34, 1189–1196 (2000)
15. Sharma, A.: Seasonal to interannual rainfall probabilistic forecasts for improved water supply management: 1 - A strategy for system predictor identification. *Journal of Hydrology* 239, 232–239 (2000)
16. Sjöberg, J., Hjalmerson, H., Ljung, L.: Neural Networks in System Identification. In: Preprints 10th IFAC symposium on SYSID, Copenhagen, Denmark, vol. 2, pp. 49–71 (1994)
17. Slini, T., Kaprara, A., Karatzas, K., Moussiopoulos, N.: PM₁₀ forecasting for Thessaloniki, Greece. *Environmental Modelling & Software* 21(4), 559–565 (2006)

How Do Attention, Intention, and Consciousness Interact?

Stephen Grossberg

Center for Adaptive Systems, Department of Cognitive and Neural Systems,
and Center of Excellence for Learning in Education, Science, and Technology,
Boston University, Boston, MA 02215
steve@bu.edu

Abstract. This talk will discuss how brain processes of attention, intention, and consciousness differ but interact. It will show how recent neural models that link brain processes to behavioral experiences clarify these interactions. The talk will, in particular, discuss some of the phenomena illustrating dissociations between attention and consciousness that were summarized in the article of Koch and Tsuchiya (2007). Supported in part by CELEST, an NSF Science of Learning Center (SBE-0354378) and by the SyNAPSE program of DARPA (HR0011-09-C-0001).

References

1. Fazl, A., Grossberg, S., Mingolla, E.: View-invariant object category learning, recognition, and search: How spatial and object attention are coordinated using surface-based attentional shrouds. *Cognitive Psychology* 58, 1–48 (2009)
2. Grossberg, S.: How does the cerebral cortex work? Development, learning, attention, and 3D vision by laminar circuits of visual cortex. *Behavioral and Cognitive Neuroscience Reviews* 2, 47–76 (2003)
3. Grossberg, S.: Resonant neural dynamics of speech perception. *Journal of Phonetics* 31, 423–445 (2003)
4. Grossberg, S.: Consciousness CLEARS the mind. *Neural Networks* 20, 1040–1053 (2007)
5. Grossberg, S.: Beta oscillations and hippocampal place cell learning during exploration of novel environments. *Hippocampus* 19, 881–885 (2009)
6. Grossberg, S., Hong, S.: A neural model of surface perception: Lightness, anchoring, and filling-in. *Spatial Vision* 19, 263–321 (2006)
7. Grossberg, S., Pilly, P.: Temporal dynamics of decision-making during motion perception in the visual cortex. *Vision Research* 48, 1345–1373 (2008)
8. Grossberg, S., Versace, M.: Spikes, synchrony, and attentive learning by laminar thalamocortical circuits. *Brain Research* 1218, 278–312 (2008)
9. Grossberg, S., Yazdanbakhsh, A., Cao, Y., Swaminathan, G.: How does binocular rivalry emerge from cortical mechanisms of 3-D vision? *Vision Research* 48, 2232–2250 (2008)
10. Huang, T.-R., Grossberg, S.: Cortical dynamics of contextually-cued attentive visual learning and search: Spatial and object evidence accumulation. *Psychological Review* (in press, 2010)
11. Koch, C., Tsuchiya, N.: Attention and consciousness: two distinct brain processes. *Trends in Cognitive Sciences* 11, 16–22 (2010)

Consciousness versus Attention

John G. Taylor

Department of Mathematics, King's College London, Strand London WC2R2LS, UK

Abstract. We consider paradigms attempting to prove that attention and consciousness are independent. As part of the analysis we introduce the notion of the ‘Creativity Effect’, and indicate how it may be used in certain experiments to explain how apparently attention-free awareness arises through a process in which attention plays a crucial role.

Keywords: Creativity; The creativity effect; Attention model; lateral spreading; argument by analogy.

1 Introduction

There is an important controversy presently being fought out in both the experiment and theory of attention: is attention necessary to be applied to a stimulus for the stimulus to be conscious? The extreme positions are that a) attention and consciousness are independent [1-6] or b) attention is always necessary for consciousness of a stimulus to arise [7, 8]. Some discussions and references supporting position b) (the majority position in brain science) are given in the papers [9 – 11]. The purpose of this brief paper is to attempt to bring together some of the arguments in this battle so that progress can be made towards a resolution of the argument.

One particularly crucial aspect in this controversy is the recognition that important aspects of mental life occur outside consciousness [12]. However we must ask as to how consciousness is then used in such partially unconscious information processing? In creativity, for example, consciousness arises as a crucial component after important unconscious processing has occurred. More generally we expect the interchange between the two states (conscious and unconscious) is highly non-trivial and needs to be teased out by experiment. It will be these presently ambiguous experiments that need further discussion in the light of this interchange.

The most important experiments, to which we will restrict our attention in this paper, have attention apparently removed by dual tasking, although awareness of stimuli still occurs. This result would imply that attention is not necessary for consciousness, so that models of attention would be of no value to furthering an understanding of consciousness. We turn to models that explore the interchange of conscious and unconscious processing more carefully, introducing the Creativity Effects. These are used to help explain the phenomena observed: attention is still a prerequisite for awareness.

2 The Basic Phenomena

The four possibilities that have to be considered [5] are: attention and separately consciousness are or are not present in a phenomenon involving a visual stimulus (where a similar analysis can be given to any modality or combination of them). As we have noted, we will concentrate here on the case of consciousness of a stimulus arising without attention apparently being applied to it. The other cases are also of interest, but do not seem to present the same crucial character of independence. Attention without consciousness has been observed in blindsight [13] so needs to be included as part of any model of consciousness arising from attention. But the phenomenon is consistent with attention being necessary for consciousness. The same is true with the absence of attention and consciousness occurring continually in reflex actions. The presence of both attention and consciousness appear to be the norm of perception [9-11].

There is an interesting list of phenomena involved with perception of stimuli without attention being paid in [5]: pop-out in search, iconic memory, gist, animal and gender detection in dual tasks and partial reportability. The most important of these tasks uses dual task methods to reduce (hopefully to zero) the attention paid to one set of stimuli appearing in the dual task by making the other task very difficult. In this way all (or a majority) of the available attention will be involved with the difficult task. Thus any awareness reported about stimuli of the second task will imply the existence of awareness of non-attended stimuli for that task.

This method has more specifically been focussed on by several recent experiments. Thus the experiments reported in [1, 2, 3] all indicate that there maybe such reduction of attention to almost zero with ability still present to carry out the dual tasks. One set of experiments have used as the second set of stimuli (with no attention claimed to be applied) switching perception arising from ambiguous figures [3]. However new data [8] has shown that attention is after all required to achieve alterations in interpretations of ambiguous figures, so negating the claims of [3]. We will specifically concentrate on [1, 2] as presenting data that, on the face of it, imply that attention is not needed for consciousness.

3 Basic Models

Models of attention do not immediately help in understanding phenomena involved in creative thinking. Such thinking involves switching from attentive to unattended processing, so leading to a greater range of neural activity than possible under attention control. Recent and past memories are easier to call upon and freer associations are able to be made between concepts as part of the unattended thinking process.

Expanded lateral spreading of neural activity, as an essential component of creativity, arises when attention is directed away from a given scene. When new concepts have thereby been activated attention needs to be redirected to determine their character in more detail. This is basic, for example, to the creative process, where a specific problem is being considered and the creative or ‘aha’ moment arises from unattended to attended thinking. This form of attention switching can be

expected to occur both quite often and can be fast, such as when attention is not being paid to a phenomenon but needs to be directed to it suddenly. A level of guesswork guided by past memories of similar situations, it is suggested, may then be used to claim what experiences had occurred during the inattentive phase. If the memories are specific enough then there is a high confidence of the guess. We termed this process the 'Creativity Effect' [11], since it involves a creative use of past memories to fill in what had been missed. However we propose that this effect occurs at two levels, higher and lower order.

Such splitting of the creativity effect is clear if we consider the results of [2], specifically the hit rates (HR) and false alarm rates (FAR), as shown in the table below. The paradigm involves two possible conditions: a stimulus is to be detected (with no time pressure, but only after a brief visual presentation) either under directed attention conditions or in the case attention is directed elsewhere (so the stimulus is 'unattended'). There are two effects to be noted when HR and FAR are compared between the two cases: firstly there is a greater HR in some trials of the unattended case, and secondly in other trials of the same paradigm there is a greater FAR for the same comparison. Thus increased accuracy can occur in some cases, whilst increased error (possibly through guessing) can occur in others. This leads to the following definition [11]:

The Higher Order Creativity Effect occurs when in an inattentive state with respect to a given stimulus and context and leads to the conscious experience of a related stimulus and/or context with a suitably high level of confidence. However the related stimulus and/or context being experienced do not need to have been in the external world, but were excited in the subject's brain through suitable lateral connections (such as in semantic concept maps or in long-term memory). This can lead to a higher FAR.

The Lower Order Creativity Effect occurs when there is an enlargement of cell receptive fields in cortical areas by bilateral spreading (without attention), leading to enhanced acuity for noisy inputs when attention is switched back. This will lead to a greater HR.

The HO Creativity Effect need not be based purely on past long-term memories, since there may be leakage from traces of working memory or iconic memories that allow some hint as to the past event of relevance. For example if a subject is performing a detection task several times there will be decaying working memories of previous detection responses, especially the most recent such response. Thus there are various internal memories that play a role in the HO and LO creativity effects:

- a) Long-term memories that are activated by the context in which the creativity effect takes place (for the HO effect);
- b) Short-term memories, as arising from slightly earlier events, especially those relevant to the present task (for the HO effect);
- c) More direct hints from the incoming visual scene, such as given only by an incomplete view which requires 'filling in' in a creative manner (for the HO effect);
- d) Enhanced acuity in lower level visual regions when noise is present and attention to the relevant regions relaxed. This would involve a shorter iconic memory to hold this activity after the stimulus has been removed (for the LO effect).

All of these internal (and possibly noisy external) stimuli may play a role in the creativity effects, so it is difficult to disentangle these sources from each other unless there is some specific feature which allows this to occur. Such would be the case, for example, if a subject is repeating a given task many times, albeit under different conditions, say of attention, between the trials. However the context and responses will all be highly relevant to any filling in needed so that the short-term memory component b) may play a preponderant role in that situation.

The creativity effects seem to lead to consciousness of the stimulus without any attention being paid to it. For the stimulus experience is reported on with confidence because it had been clearly seen. Such was the case in a recent personal experience of mine, when I was very sure I had ‘seen’ a particular state of affairs in my house but a closer look showed that was not true and a quite different state of affairs was actually in place. The situation in my and many other cases would be that of little or no attention being paid to the relevant and actual stimulus, and a completely different stimulus ‘viewed’. This appears to be a clear case when consciousness is created outside attention.

However it is soon seen that this claim is false in my case, the stimulus being an open bathroom window (or so I thought). I was indeed highly conscious of the open bathroom window that I had hallucinated as being present from my brief glimpse into my bathroom. But I was not conscious of the actual stimulus present in the visual scene, which was a closed bathroom window. I was only conscious of what clearly was an imagination or hallucination inside my own head. Moreover I was attending very strongly to that hallucination. Thus I was conscious of an internally-generated stimulus to which I was attending (and planning my response). Not consciousness without attention once the hallucination had occurred, but then guided by attention to elicit consciousness. This also occurs with a high error rate.

Such is very likely the case in many other higher order creativity effect situations: an internally-generated image arises in a given situation, there is both attention and consciousness of it, but there is no external stimulus at all of which there is consciousness. The lower order creativity effect is quite the opposite, granting greater perceptual acuity to the viewer. But again in this case the lateral spreading of activity, producing a larger field of view, will ultimately lead to the conscious perception of the viewed percept.

In the creativity effects, then, attention is released to allow for analogical reasoning that produces associations to what was weakly attended to so as to allow for consciously filling in gaps in knowledge associated with the weakly attended stimuli. It can arise either by ‘hallucinating’ as in the HO Creativity Effect (so increasing the FAR) or by increased sensitivity to iconic memory traces of a stimulus, as in the LO Creativity Effect, so increasing the HR and making the processing more efficient.

4 Paradigm Analyses

We now turn to analyze the results of the experiment of [2] in terms of the creativity effect. The paradigm uses the original Posner cueing paradigm to direct attention to the right side of space. Any target appearing on the left side will thus be under lowered attention. Moreover the paradigm used an arrow precue (on for 500 ms), then

a stimulus on for 367 ms, and finally a post-cue arrow, but with no stimulus, until an answer was obtained. So in both the invalid and valid cases the stimulus was no longer present when the target detection process commenced, with redirection of attention being needed in the invalid case.

The main result of the experiment is that in the invalid case there are a surprisingly large number of target hits, as well as false alarms. Thus it is quoted in the paper that there were 33% ‘target present’ responses in the strongly attended case, whereas under similar conditions, but now with weak attention, there were 56% ‘grating present’ responses. It is necessary to ask what stimuli the responders were seeing in the at least 23% of responses in the weak attention case when there was no target grid. And why had the responders been so confident that there had been a target present?

A more detailed set of responses, occurring for the four different levels of grid contrast reported on in [2]. These are (private communication, Rahnev, 2009):

Table 1. Table of HR and FAR for cued (attended) and uncued (unattended) target detection (where HR = hit rate, FAR = false alarm rate). Noise added to the target increases from right to left in the table.

Parameter				
HR(cued)	0.37	0.60	0.67	0.92
HR(uncued)	0.76	0.79	0.91	0.91
FAR(cued)	0.27	0.21	0.11	0.14
FAR(uncued)	0.74	0.54	0.26	0.40

From the table the HR(cued) decreases faster as noise increases than does HR(uncued). Thus for the case of the most difficult grating angle difference, in the cued case, there are 2/9 fewer hits/misses as for the uncued case; there are also 1/9 fewer false +ve as compared to true –ve cases. This justifies the claim that there are far more correct target recognitions but also more incorrect ‘guesses’ in the uncued as compared to the cued case.

In particular it is the case that there is greater accuracy in its detection when a grid is present in the uncued as compared to the cued case, so the lack of attention is allowing, as noted earlier, a larger field of view (the LO creativity effect). On the other hand when there is no grid present we can suspect that there is greater creativity by guessing in the uncued rather than cued case, with the lack of attention producing less accurate results (but obtained with greater confidence) in the former than latter case (HO creativity effect). Thus both aspects of attention reduction help to explain the results of this paradigm: that of an enlarged field of view in lower cortices (when a grid is present, so increased accuracy by the LO creativity effect) as well as the HO creativity effect of hallucinating (when no grid is present) with greater error and confidence.

Such an explanation is in agreement with the further results of [2], which show by suitable manipulation of the data that the decisions of their subjects could only arise from at least a two channel system: “Thus we reject the hypothesis that the attended and unattended trials are managed by the same decision making system” [2, p39]. This would be so if the two different creativity effects were being employed as described above, with the LO using unattended processing, the HO attended processing.

We conclude that the data of [2] do not support the claim of the authors that there is only independence of consciousness and attention. Rather the results emphasize the need to consider in what manner the paradigm needs more careful analysis associated with the creativity effects.

The creativity effects explanation of the above results leads to several predictions:

- 1) The expectation that false positives will especially arise in a weakly attended task after a run of strongly attended cases, when detection had been especially strong. For then there will be a strong bias from the working memory containing an image of the grid being present.
- 2) There should be a decreasing level of false positives in a sequence of unattended stimuli, as the bias from working memory of an observed grid decays away.
- 3) In a sequence of alternations of presentations of unattended and attended grids there should be a relatively constant level of false positives from the relatively constant level of bias from the working memory for the attended grids in the strongly attended cases.
- 4) Comparing sequences of form AU, AAU, AAAU, AAAAU, AAAAAU, with increasing sequences of AAA...A trials followed by a U trial, there should be increasing numbers of false positives in the U trials as there is increasing bias from the lengthening working memory as the sequence of A trials increases (although this effect may soon run out of steam as earlier and earlier A trials have increasingly smaller effect on the bias of trial responses).
- 5) Such a creativity effect explanation should be checked by brain imaging, especially by fMRI. In the case of 4) above there should be an increase in the activity of a relevant working memory site as the length of the AA...A sequences increases. There should also be an important correlation (and associated causal flow of information) shown between the relevant working memory site and a decision-making site in cingulate cortex.
- 6) There is still the open question of switching between use of higher acuity unattended processing and hallucinations.

Similar conclusions can be reached for the data presented in [1]. In the paradigm subject were presented with a central arrow cue pointing either up or down for 500 msec. 75% of the time they had a valid cue to a slanted grid whose direction slant had to be reported; 25% of the time the cue was invalid with a grid in the opposite vertical direction being presented. A mask was then presented after a further 500msecs. Subjects had to indicate whether the patch, appearing either at a cued or a noncued location, was tilted to the left or to the right and indicate their confidence in this judgment.

There is again a dearth of raw data, but the figures 3 & 7 in [1] are indicative of certain structure. These figures indicate a close similarity of the mean confidence as a function of the correct response across the two conditions of high and low attention to the stimulus (valid or invalid cues). Firstly there is a natural break in both sets of figures between correct response levels of 60 – 80%. For the lower correct response levels (those below 60%) there are seen to be more points involving low attention than high attention ones (where this is taken over the different levels of difficulty employed in the paradigm). The natural break observed can be conjectured as arising from the HO creativity effect being used at the lower correct response levels (for both

high and low attention cases). For the higher correct response points (for both low and high attention) it can be conjectured that there is enough attention in the low attention condition so as to provide about as much information as contained in the high attention condition. This explains the agreement with the results in figure 7. Those in figure 3 of [1] are slightly different in that the lower portion of the two curves (for high and low attention conditions respectively) is roughly horizontal. Again however this provides a natural break with the high correct response points, so a similar explanation can be conjectured: the lower response accuracy points of both high and low attention conditions arise from the HO creativity effects whilst those for the higher accuracy response values arise from increasingly noise-free attended processing.

5 Conclusions

The most important conclusion of this analysis is that certain psychological paradigms must be treated with care in their probing of the relation between attention, confidence and consciousness. In particular a) the relation between attention and consciousness is still consistent with the necessity of attention for consciousness b) the relation between confidence and consciousness must be treated with care, in paradigms such as those of [1] or [2]. If one or other of the creativity effects proposed in the paper are being employed then there need be no relation at all between confidence and consciousness if attention to a given stimulus is low. Nor is the supposed weakness of the ‘weak’ attention condition relevant to the amount of attention then applied to the creatively produced illusions of the external world, although weak attention is needed to allow the HO creativity effect to arise in the first place and lead to the imagined world with high confidence, or the more efficiently probed world in the noisy case (the LO creativity effect), as reported in [2].

Acknowledgements

I would like to thank Prof Lau and Dr Rahnev for a copy of their paper (Rahnev et al, 2009) before publication as well as for helpful communications concerning other aspects of the subject and further data. Thanks also are due to Prof Pastukhov for useful interactions.

References

- [1] Willimzig, C., Tsuchiya, N., Fahle, M., Einhäuser, W., Koch, C.: Spatial attention increases performance but not subjective confidence in a discrimination task. *Journal of Vision* 8(5), 1–10 (2008)
- [2] Rahnev, A., Maniscalco, B., Huang, E., Bahdo, L., Lau, H.: Weakly attended stimuli produce an inflated sense of subjective visibility (U. of Columbia preprint) (2009)
- [3] Pastukhov, A., Braun, J.: Perceptual reversals need no prompting from attention. *Journal of Vision* 7(10), 5:1–17 (2007)
- [4] Tsuchiya, N., Koch, C.: Attention and consciousness. *Scholarpedia* 3(5), 4173 (2007)

- [5] Koch, C., Tsuchiya, N.: Attention and consciousness: two distinct brain processes? *Trends in Cognitive Sciences* 11(1), 61–62 (2007)
- [6] Lamme, V.A.F.: Separate neural definitions of visual consciousness and visual attention: a case for phenomenal awareness. *Neural Networks* 17(5-6), 861–872 (2004)
- [7] Srinivasan, N.: Interdependence of attention and consciousness. In: Banarjee, R., Chakrabarti, B.K. (eds.) *Progress in Brain Research*, ch. 6, p. 168 (2008)
- [8] Zhang, P., Engel, S., Rios, C., He, B., He, S.: Binocular rivalry requires visual attention: Evidence from EEG. *Journal of Vision* 9(8), 291a (2009); Abstract 291
- [9] Taylor, J.G.: A Review of Models of Consciousness. In: Cutsuridis, V., Taylor, J.G. (eds.) *The Perception and Action Cycle*. Springer, Berlin (2010)
- [10] Taylor, J.G.: CODAM: A Model of Attention Leading to the Creation of Consciousness. *Scholarpedia* 2(11), 1598 (2007)
- [11] Taylor, J.G.: The Creativity Effect: Consciousness versus Attention. In: IJCNN 2010 (in press, 2010)
- [12] Wilson, T.D.: *Strangers to Ourselves*. Belknap, Harvard University Press, Cambridge, MA (2002)
- [13] Kentridge, R.W., Heywood, C.A., Weiskrantz, L.: Spatial attention speeds discrimination without awareness in blindsight. *Neuropsychologia* 42, 831–835 (2004)

On the Fringe of Awareness: The Glance-Look Model of Attention-Emotion Interactions

Li Su¹, Philip Barnard¹, and Howard Bowman²

¹ MRC Cognition and Brain Sciences Unit, Cambridge, United Kingdom

{Li.Su,Phil.Barnard}@mrc-cbu.cam.ac.uk

² Centre for Cognitive Neuroscience and Cognitive Systems

and School of Computing, University of Kent, Canterbury, United Kingdom

H.Bowman@kent.ac.uk

Abstract. In previous work, we have developed a “Glance-Look” model, which has replicated a broad profile of data on the semantic Attentional Blink (AB) task and characterized how attention deployment is modulated by emotion. The model relies on a distinction between two levels of meaning: implicational and propositional, which are supported by two corresponding mental subsystems. The somatic contribution of emotional effects is modeled by an additional body-state subsystem. The interaction among these three subsystems enables attention to oscillate between them. Using this model, we have predicted the pattern of conscious perception during the AB and the changes of awareness when emotional or other task irrelevant processing occurs. We provide a specific account of the interaction between attention, emotion and consciousness. In particular, the dynamics of two modes of attending to meaning (implicational being more distributed and propositional being evaluative and specific) give rise to fringe awareness.

Keywords: fringe awareness, Attentional Blink, emotion, Glance-Look model, consciousness, body-state.

1 Introduction

The relationship between consciousness and attention is one of the most controversial issues in psychology, philosophy and cognitive neuroscience. One key debate is centered on whether or not attention and consciousness can be untangled into separate processes. A paradigm called the Attentional Blink (AB) [1] has demonstrated a situation where the same sensory input can lead to both conscious and non-conscious perception. The task involves targets being presented using Rapid Serial Visual Presentation (RSVP) at around ten items a second. The identification of a second target is impaired when it is closely preceded in time (< 500ms) by a first target. In addition, findings suggest that attention and consciousness are not only separable, but may also oppose each other. For instance, it has been discovered that task manipulations and emotional states can attenuate blink effects (i.e. enhance the awareness of the second target) by, it is argued, encouraging a more distributed state of attention, e.g. by using music, positive affect or dynamic visual patterns to

counteract an overinvestment of attention [2,3,4]. Thus, in this context, reducing attentional focus seems to improve awareness. Hence, the AB paradigm provides a solid platform for investigation of the relationship between attention, emotion and conscious perception. This paper shows how our previously developed “Glance-Look” model [5,6,7] can also provide a general information processing account for the interactions between attention, emotion, consciousness, and fringe awareness, as well as, more specifically, explaining overinvestment findings in the AB.

2 The “Glance-Look” Model

2.1 Key-Distractor Attentional Blink Task

The “Glance-Look” model, as shown in Figure 1, was first proposed to explain a variant of the AB paradigm, in which words were presented at fixation in RSVP format and targets were only distinguishable from background items in terms of their meaning. Participants were simply asked to report a word if it refers to a job or profession, such as “waitress”, and these targets were embedded in a list of background words that all belonged to the same category, e.g. nature words. Participants could report what they believed was the target word’s identity (Correct ID), say “Yes” if they were confident a job word had been there, but could not say exactly what it was (to capture a degree of awareness of meaning), or say “No” if they did not see a target, and there were, of course, trials on which no target was presented. However, streams also contained a key-distractor item, which, although not in the target category, was semantically related to that category, such as “tourist” or “husband” [8], or emotionally charged, such as “cancer” [9]. The serial-position that the target appeared after the key-distractor was varied. The effect of attentional capture is encapsulated in the serial position curve in Figure 2(A). We call this the key-distractor AB task.

2.2 Basic Structure of the “Glance-Look” Model

The key principles that underlie the “Glance-Look” model are as follows. 1) Items are composed of several constituent representations (CRs) passed through a pipeline from the visual system to the response system. On every cycle, a new CR enters the pipeline and all CRs currently in transit are pushed along one place. In this sense, it could be viewed as an analogue of a sequence of layers in a neural network, e.g. a synfire chain [10]. 2) The processing of meaning is divided into two stages, which are supported by the implicational (*Implic*) and propositional (*Prop*) subsystems. Each subsystem assesses a different type of meaning using Latent Semantic Analysis [11], a statistical learning technique, related to Principle Component Analysis and Hebbian learning. 3) A body-state subsystem monitors the outputs of Implic and feeds back emotional information in the form of “somatic markers” [12]. 4) Only one subsystem can be attended at a time, and it is only when attention is engaged at a subsystem that it can assess the salience of items passing through it. All unattended subsystems in the model process stimuli in parallel, in a kind of “zombie” mode, similar to that suggested by Crick and Koch [13]. We argue that it is attention and the interaction between these subsystems that brings information into consciousness.

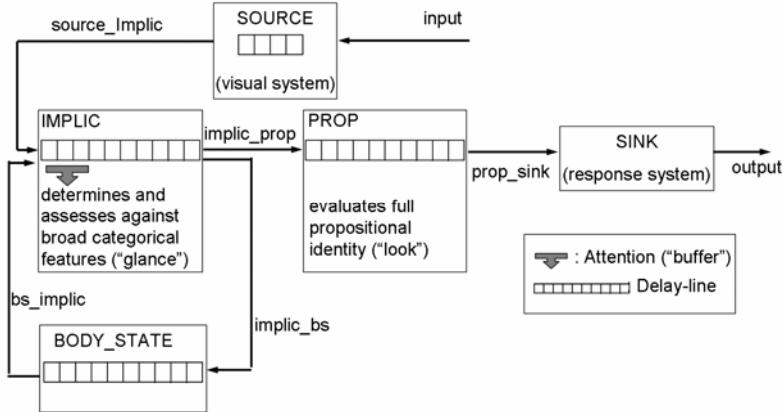


Fig. 1. The top-level schematic diagram of the “Glance-Look” model. Names in uppercase are processes, e.g. **IMPLIC**, names in lowercase are communication channels, e.g. **implic_prop**. **IMPLIC**, **PROP**, **BODY_STATE**, **SOURCE**, and **SINK** represent implicational subsystem, propositional subsystem, body-state subsystem, visual system, and response system respectively.

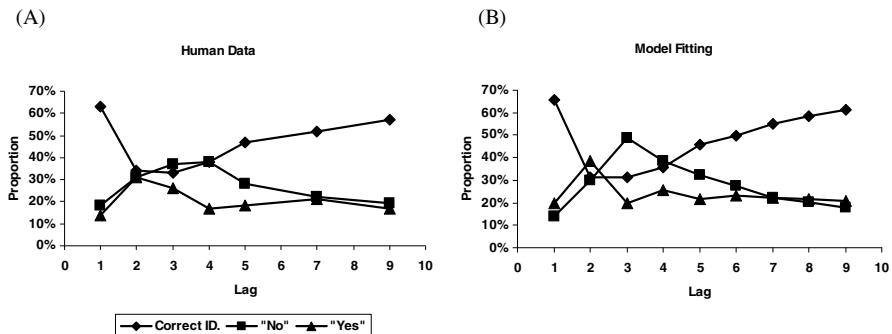


Fig. 2. (A) Proportion of different types of responses from humans [8]. Lag indicates the number of items intervening between the key-distractor and the target. (B) Model simulates the behavioral curves of the AB [5].

2.3 How the Model Blinks

In the context of the AB, attention is captured when a key-distractor is interpreted as implicationally salient, as it is task or personally relevant. This then causes attention to be redeployed to Prop, in order to enable a more detailed assessment of the salience of the key-distractor. This redeployment of attention leaves a temporal window in which implicational salience is not fully assessed. In particular, the meaning of a target word can be processed to three different degrees, which, we argue, reflect

different degrees of awareness and thus lead to three types of response, one of which provides a computationally explicit characterization of fringe awareness. Words that are fully (i.e. both implicational and propositionally) interpreted can be (fully) consciously accessed and reported correctly with their identity. Some targets can be only partially processed, i.e. insufficient CRs have been assessed (by Prop), resulting in fringe awareness, reflecting the “Yes” response. This only happens during the transition between complete conscious and unconscious states. As shown in Figure 2, the results of the model simulation were comparable to human performance, with fringe awareness particularly evident at lag 2. Finally, some targets can be implicational totally un-interpreted reflecting complete unawareness of the presence of target words, i.e. the “No” responses. This state is most likely to occur around the deepest points in the blink.

2.4 The “Glance-Look” Account of the Attenuation Effect

As previously discussed, there is evident that distracting participants can reduce blink depth. Consequently, overinvestment theories of the AB have become prominent. The degree of, distraction-induced, attenuation reported in [2,3,4] should, though, reflect two factors: the degree to which the ancillary task has direct consequences for the representation of generic (implicational) meaning and the extent to which the reporting of an item requires extended evaluation of propositional meanings in our second stage. To elaborate further, in a typical laboratory setting, participants are encouraged to recall as accurately as possible. As previously argued by several authors [2,3,4], this could well result in rather more investment of attention than is strictly necessary to accomplish item report. This hypothesis is consistent with our model, in the sense that, overinvestment may reflect extended processing in our second “propositional” stage, where attention may take a long time to switch back to a state in which implicational representations are attended. The implicational mode of attending to meaning has a broader focus on generic meaning, which may incorporate affect, and derivatives of multimodal or lower order inputs, such as music. When subjects are exposed to dynamic patterns, being visual, musical or internally generated, while performing the central AB task, there would be more changes in input to implicational meaning. With our model of distributed control, these may well encourage the implicational mode of attending to meaning, and support more distributed awareness of this type of generic meaning. Crucially, the paradigm often involves reporting letters in a background stream of digits [2,3,4]. Letters are drawn from a small and highly familiar set, and hence, in the limit, this may require only the briefest “look” at a propositional representation to support correct report.

Attenuation should be less pronounced either with secondary tasks whose content does not directly influence the level of generic (implicational) meaning or, as with semantic blink effects, where a fuller evaluation of propositional meanings is required. Should such effects be found, it would provide an encouraging convergence between basic laboratory tasks and the literature on attention to meaning and affect in emotional disorders, using a non-computationally specified version of our current proposal [14].

3 Modeling Conscious Perception during the AB

The “Glance-Look” model takes specific account of the interaction between attention, emotion and consciousness. Firstly, we have shown that lack of awareness can be accounted for by the allocation of attention to different levels of meaning in a system where there is only distributed control of processing activity. Just as the focus of our attention may shift among entities in our visual and auditory scenery under the guidance of salient change, shifts in attention to different entities in our semantic scenery can lead to RSVP targets being either, 1) correctly identified; 2) giving rise to a fringe awareness of presence; or 3) overlooked. Salience states at each of two levels of meaning allow these three response patterns to be captured. Although the proposal, like that of Chun and Potter [15], relies on two stages, both of our stages are semantic in nature and the temporal dynamic involves controlled changes in the focus of attention, rather than classic capacity or resource limitations. The idea of monitoring a generic form of meaning for implicational salience, the level at which affect is represented in the model, and switching only when required to evaluate propositional meaning, represent two “modes” of attending to meaning. The former mode has a broader focus on generic meaning (i.e. the “gist”) and the latter a more evaluative focus on specific meanings, which can be verbally reported. This is similar to the distinction in the literature between “phenomenal” and “access” awareness [16]. Furthermore, the broader mode of processing meaning bears some resemblance to recent suggestions that task manipulations can attenuate blink effects, by encouraging a more distributed state of awareness, which would arise at our implicational level. In particular, music, positive affect or dynamic visual patterns may counteract the overinvestment of attention [2,3,4] and produce a fleeting conscious percept [13].

4 Conclusions

In summary, consciousness is modeled as an emergent property from the interaction among three subsystems: implicational, propositional and body-state. In particular, we differentiate two types of consciousness. One is akin to full “access” awareness, i.e. conscious content can be verbally reported, and is supported by both implicational and propositional processing. In other words, it is a result of a detailed “look” and more extensive mental processing. The other is akin to “phenomenal” (or fringe) awareness, and lacks the capacity of linguistic access. We argue that the latter is a result of attending to the implicational level or “glance”. It is also notable that the implicational level is holistic, abstract and schematic, and is where multimodal inputs are integrated, and affect is represented and experienced [17].

In addition, the “Glance-Look” model makes several predictions on the relationship between these two modes of consciousness. First, fringe awareness provides a basis for a more complete state of consciousness. Second, comparing to full access awareness, phenomenal or fringe awareness is directly affected by emotional, multimodal, body-state and lower order inputs. However, once propositional level information has been attended, a conscious percept is much less likely to be interrupted. The validation of these predictions awaits further experimental work.

References

1. Raymond, J.E., Shapiro, K.L., Arnell, K.M.: Temporary suppression of visual processing in an RSVP task: an attentional blink. *Journal of Experimental Psychology: HPP* 18(3), 849–860 (1992)
2. Arend, I., Johnston, S., Shapiro, K.: Task irrelevant visual motion and flicker attenuate the attentional blink. *Psychonomic Bulletin and Review* 13(3), 600–607 (2006)
3. Olivers, C.N.L., Nieuwenhuis, S.: The beneficial effects of concurrent task: irrelevant mental activity on temporal attention. *Psychological Science* 16, 265–269 (2005)
4. Olivers, C.N.L., Nieuwenhuis, S.: The beneficial effects of additional task load, positive affect, and instruction on the attentional blink. *Journal of Experimental Psychology* 32, 364–379 (2006)
5. Su, L., Bowman, H., Barnard, P.J.: Attentional Capture by Meaning, a Multi-level Modelling Study. In: 29th Annual Meeting of the Cognitive Science Society, pp. 1521–1526 (2007)
6. Su, L., Bowman, H., Barnard, P.J., Wyble, B.: Process algebraic modelling of attentional capture and human electrophysiology in interactive systems. *Formal Aspects of Computing* 21(6), 513–539 (2009)
7. Bowman, H., Su, L., Wyble, B., Barnard, P.J.: Salience Sensitive Control, Temporal Attention and Stimulus-Rich Reactive Interfaces. In: *Human Attention in Digital Environments*. Cambridge University Press, Cambridge (2010)
8. Barnard, P.J., Scott, S., Taylor, J., May, J., Knightley, W.: Paying attention to meaning. *Psychological Science* 15(3), 179–186 (2004)
9. Barnard, P.J., Ramponi, C., Battye, G., Mackintosh, B.: Anxiety and the deployment of visual attention over time. *Visual Cognition* 12(1), 181–211 (2005)
10. Abeles, M., Bergman, H., Margalis, E., Vaadia, E.: Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of Neurophysiology* 70, 1629–1638 (1993)
11. Landauer, T.K., Dumais, S.T.: A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction and Representation of Knowledge. *Psychological Review* 104, 211–240 (1997)
12. Damasio, A.: *Descartes' Error*. G.P. Putnam's Sons, New York (1994)
13. Crick, F., Koch, C.: A framework for consciousness. *Nat. Neur.* 6, 119–126 (2003)
14. Teasdale, J.D.: Emotional processing, three modes of mind and the prevention of relapse in depression. *Behaviour Research and Therapy* 37, S53–S77 (1999)
15. Chun, M.M., Potter, M.C.: A two-stage model for multiple target detection in rapid serial visual presentation. *Journal of Experimental Psychology: HPP* 21(1), 109–127 (1995)
16. Lamme, V.A.: Why Visual Attention and Awareness are Different. *Trends in Cognitive Science* 7, 12–18 (2003)
17. Barnard, P.J.: Interacting cognitive subsystems: modelling working memory phenomena with a multi-processor architecture. In: Miyake, A., Shah, P. (eds.) *Models of Working Memory*, pp. 298–339. Cambridge University Press, Cambridge (1999)

No Stopping and No Slowing: Removing Visual Attention with No Effect on Reversals of Phenomenal Appearance

Alexander Pastukhov, Victoria Vonau, and Jochen Braun

Cognitive biology, Otto-von-Guericke Universität, Leipziger Strasse 44 / Haus 91,
39120 Magdeburg, Germany

{Alexander.Pastukhov, Jochen.Braun}@ovgu.de,
Victoria.Vonau@st.ovgu.de,
<http://kobi.nat.uni-magdeburg.de>

Abstract. We investigated whether visual selective attention contributes to the reversals of phenomenal appearance that characterize multi-stable displays. We employed a rotating-ring display that reverses appearance only at certain phases of its rotation (i.e., when in full-frontal view). During this critical window of time, observers were required to perform a shape discrimination task, thus diverting attention from the rotating ring. Our results showed that perceptual reversals were neither stopped nor slowed by this manipulation. In contrast, interrupting the display during the critical times increased the frequency of perceptual alternations significantly. Our results go beyond earlier findings that sustained withdrawal of attention slows, but does not stop, perceptual reversals. Accordingly, the available evidence strongly suggests that visual selective attention plays no causal role in multi-stable perception.

Keywords: multi-stable perception, attention.

1 Introduction

When faced with ambiguous input, the visual system switches spontaneously between alternative interpretations. Although first scientific study of multi-stable perception was performed nearly two centuries ago [1], the mechanisms behind phenomenal alternations are still not fully understood. One theory, which was the first theory to have been put forward historically and which retains many adherents to this day, is that reversals of phenomenal appearance simply reflect shifts of visual selective attention [2,3]. Indeed, considerable evidence has accumulated over the years that both endogenous [4,5] and exogenous [6,7] manipulations of visual attention profoundly modulate the dynamics of multi-stable appearance. In addition, studies with functional imaging and neurophysiological techniques find consistently that reversals of phenomenal appearance are accompanied by activity in parietal and frontal cortical areas that are associated with visual attention and working memory (for a review, see [8]). However, it seems equally possible that visual attention triggers reversals or that reversals attract attention, so that the causal relation between attention and reversals remains unclear.

Several studies have addressed this issue with dual-task paradigms, requiring observers to attend to a demanding discrimination task while also reporting on the phenomenal appearance of a multi-stable display [9,10,11]. Such a sustained and partial withdrawal of attention from multi-stable display significantly slows the pace of phenomenal reversals but fails to eliminate them. In fact, withdrawing attention has a similar effect as reducing stimulus intensity [10,11]. Other studies have attempted to withdraw attention completely from multi-stable displays by requiring observers to report only (or mostly) on the attention-engaging task. To monitor phenomenal appearance, these studies either relied on occasional observer reports [11] or on physiological correlates of the dominant percept such as eye movements [12] or EEG modulations [13]. The former two studies found that, in the absence of visual attention, phenomenal reversals continued, while the latter study seems to suggest that reversals may cease altogether.

To further investigate the causal role of visual attention in multi-stable perception, we have devised a modified version of the kinetic-depth display. The phenomenal appearance (rotation in depth) of this display reverses almost exclusively at a particular phase of the rotation, specifically, when the stimulus is in full-frontal view. To document this, we performed a control experiment in which observers reported phenomenal reversals together with the position of a “indicator dot” at the moment of reversal. (The “indicator dot” revolved around the display like the second hand of an analog clock.) All subjects reported reversals to occur shortly after the rotating ring passed the 0° (“full-frontal”) orientation (Fig. 1A, mean switch time 29 ± 5 ms). 98% of switches occurred within ± 150 ms of 0° orientation. Taking decisional bias into account [14], it seems likely that the vast majority of reversals occurred within this critical window of time.

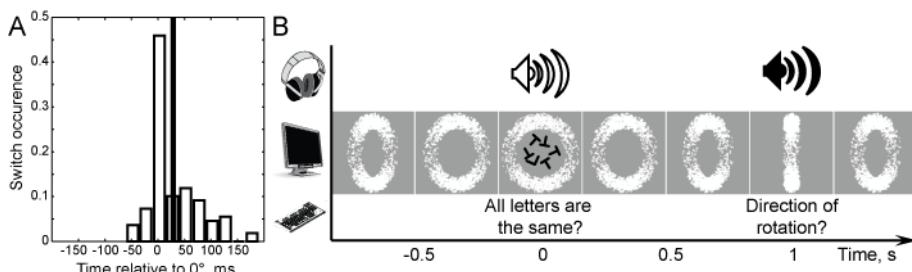


Fig. 1. A) Distribution of the reported switch times relative to the 0° (full-frontal view) orientation of the ring (time 0). Solid line shows mean of the distribution. B) Schematic procedure of the experiment. An ambiguously rotating ring was presented continuously but observers were prompted by an auditory signal (top row, filled icon) to report the direction of rotation only once a second. An attention-demanding letters task was presented for 300 ms and observers were prompted by an auditory single (top row, open icon) to report whether or not all letters were identical. For single task conditions observers were prompted to report only on one task.

In contrast with previous studies, which kept sustained attention on the central task for the entire duration of the block, we engaged attention only intermittently, specifically, for 300 ms around the 0° orientation ($[-150\ldots 150]$ ms, 15% of total rotation time, see Fig. 1B). If visual attention is required to trigger a phenomenal reversal,

then withdrawing it from the rotating ring during the critical windows of time should stabilize phenomenal appearance and reduce the number of reversals. On the other hand, if withdrawing attention modulates merely the effective contrast of the display, we would expect at most a minor (<10%) decrease of the reversal rate [11].

2 Results

To ensure that all attention was directed away from the rotating ring, we have used a letter discrimination task, which has been shown to be highly attention demanding [15,16]. Observers were prompted by an auditory signal to respond whether all letters in the array were the same (Fig. 2B). They were also prompted, by a different tone, to respond on the ring's direction of rotation when the ring was around 90° orientation. This ensured that observers payed attention exclusively to one task at a time, rather than splitting attention. In single task conditions observers responded to one of the tasks, while in the dual task situation they responded to each task in turn.

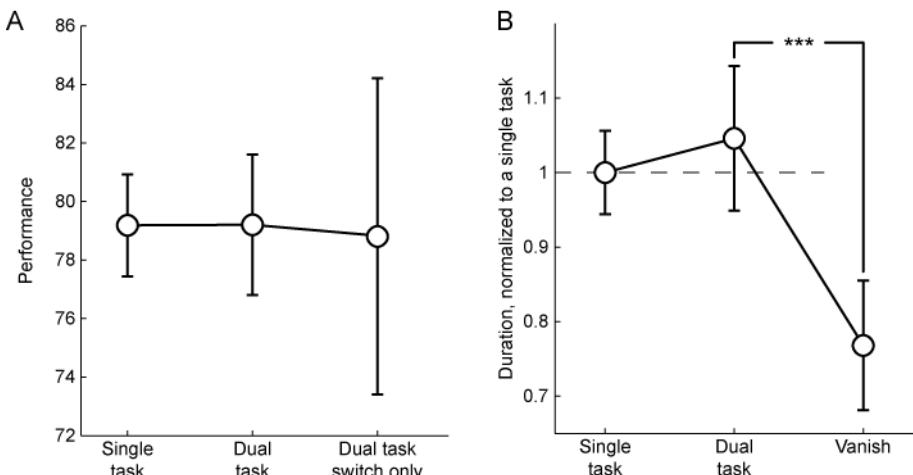


Fig. 2. Results. A) Central task average block performance for single and dual tasks, and average performance only during trials with a reported perceptual switch (dual task condition only, mean \pm 95% confidence interval). B) Mean dominance duration for *single task*, *dual task* and *vanish* condition. Durations are normalized to the mean dominance duration during single task condition of the session.

Observer's performance on the central task shows that they paid full attention to the central task in the dual task condition (Fig. 2A). Critically, there is no evidence that switches occurred only when observers failed to focus their attention on letters: their performance on trials with reported switches is not significantly different from their single task performance (binomial distribution, $p=0.93$).

In contrast with previous studies, which observed slowing down of perceptual alternations if attention was withdrawn for the entire block duration, we find no significant increase in mean dominance duration during the dual task condition (see Fig. 2,

increase of 4.3%, Wilcoxon ranksum test, $p= 0.16$). We have also examined whether absence of attention on multi-stable displays is equivalent to its physical absence. In a control condition (*vanish*) rotating ring was removed from the screen during the central task target array presentation. Unlike withdrawal of attention, physical absence caused a significant increase of the alternation rate: mean dominance duration was smaller by 21% compared to the single task (Wilcoxon ranksum test $p<0.001$ then comparing with both single and dual task conditions).

3 Discussion

In the present study, we have examined the effect of withdrawing attention during the critical windows of time in which an ambiguously rotating ring may reverse its phenomenal appearance. Unlike previous studies, which found that sustained withdrawal of attention slows down the pace of phenomenal reversals, we observed that the intermittent withdrawal of attention had only a minor and non-significant effect on multi-stable dynamics (4.3% fewer reversals). This essentially negative result was obtained in spite of the well-documented effectiveness of the attention-engaging task [15,16]. This marginal and non-significant reduction in the number of reversals is consistent with the idea that withdrawing attention merely reduces the effective contrast of the display. On the basis of previous results [11], we estimated that a reduction of effective contrast should decrease the number of reversals by than 10%. A physical interruption of the stimulus produces a very different effect and increases the number of reversals by 21%, consistent with previous observations [17].

In a recent preliminary report [13], Zhang and colleagues employed frequency-tagging to monitor the phenomenal appearance of a multi-stable display by way of evoked scalp-potentials (EEG). They report that, when visual attention was withdrawn from the display, the frequency-tagging failed and that it was no longer possible to establish a dominant percept. From this the authors infer that neither percept dominated phenomenal appearance and that perceptual reversals ceased. However, other interpretations are possible as well. For example, withdrawing attention may have reduced effective contrast (and the amplitude of scalp potentials) to the point that phenomenal appearance could no longer be monitored in this way. Alternatively, as mentioned above, attention acts as an “effective contrast”, boosting strength of the underlying neural responses [18,19]. Its withdrawal would decrease response amplitudes, particularly in the presence of a central task [20]. This reduction could render the display simply too weak to induce binocular rivalry, which is well known to require medium to high contrast [21]. Yet another possibility is that the flickering presentation required for frequency modulation could have rendered the display more prone to fusion. Accordingly, it remains to be seen whether this intriguing finding with an exotic display [13] can be generalized to more prototypical multi-stable situations.

We find that withdrawing attention intermittently during the time of phenomenal reversals has essentially no effect on the time course of multi-stable perception. This observation confirms and extends the conclusion of several earlier studies, which suggest that attention merely modulates the effective contrast of the stimulus without taking any causal part in perceptual reversals. We concluded that perceptual reversals are spontaneous and do not require prompting by visual attention.

4 Methods

Three participants (including second author) took part in the experiment. Procedures were approved by the medical ethics board of the University of Magdeburg and informed consent was obtained from all observers. Stimuli were displayed on a 21" CRT screen: 1600x1200 pixels, 100 Hz refresh rate, viewing distance was 70 cm.

Ring stimulus (diameter 12.5°, rotation speed 0.25Hz) consisted an orthographic projection of 2000 dots (diameter 0.2°) on the surface of the ring. Observers were prompted by a low frequency tone auditory signal (200 ms before ring reached position then it is orthogonal to the viewer) to report with a right hand its direction of rotation (left, right or unclear). Letter task consisted of an array of randomly rotated letters ('T' or 'L', size: 2°), presented for 300 ms (starting time 150 ms before the ring reached position when it is 'flat' for the viewer) around the fixation (eccentricity 8.5°). Target array was immediately followed by a mask, consisting of the same number of randomly rotated letters 'F' (200 ms). Number of letters was manipulated to ensure ~80% performance in a single task *letters* condition. Observers were prompted to report with a left hand whether odd item was present by a high frequency tone (200 ms before the target array onset). Visual stimulation was identical, but observers were prompted to report only rotation, only letter task or both intermittently. During *vanish* condition ring stimulus was absent during letter task target arrays presentation (300 ms). In control experiment conducted to establish time of the switch, observers viewed a continuously rotating ring with additional "indicator dot" revolving around the display like the second hand of an analog clock [14]. Observers were instructed to report perceptual switch by pressing 'space' button, while memorizing position of the indicator dot. After that they have manually adjusted its location to match its location at the time of the switch.

References

- [1] Wheatstone, C.: Contributions to the physiology of vision—part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision. Philosophical Transactions of the Royal Society of London 128, 371–394 (1838)
- [2] von Helmholtz, H.: Treatise on Physiological Optics, vol. 3. The Optical Society of America, Birmingham (1866)
- [3] Leopold, D.A., Logothetis, N.K.: Multistable phenomena: changing views in perception. Trends. Cogn. Sci. 3, 254–264 (1999)
- [4] Meng, M., Tong, F.: Can attention selectively bias bistable perception? Differences between binocular rivalry and ambiguous figures. Journal of Vision 4, 539–551 (2004)
- [5] van Ee, R., van Dam, L.C., Brouwer, G.J.: Voluntary control and the dynamics of perceptual bi-stability. Vision Res. 45, 41–55 (2005)
- [6] Chong, S.C., Tadin, D., Blake, R.: Endogenous attention prolongs dominance durations in binocular rivalry. Journal of Vision 5, 1004–1012 (2005)
- [7] Mitchell, J.F., Stoner, G.R., Reynolds, J.H.: Object-based attention determines dominance in binocular rivalry. Nature 429, 410–413 (2004)
- [8] Sterzer, P., Kleinschmidt, A., Rees, G.: The neural bases of multistable perception. Trends in Cognitive Sciences 13, 310–318 (2009)

- [9] Reisberg, D., O'Shaughnessy, M.: Diverting subjects' concentration slows figural reversals. *Perception* 13, 461–468 (1984)
- [10] Paffen, C.L., Alais, D., Verstraten, F.A.: Attention speeds binocular rivalry. *Psychological Science: a Journal of the American Psychological Society, APS* 17, 752–756 (2006)
- [11] Pastukhov, A., Braun, J.: Perceptual reversals need no prompting by attention. *Journal of Vision* 7, 5 (2007)
- [12] Leopold, D.A., Fitzgibbons, J.C., Logothetis, N.K.: The Role of Attention in Binocular Rivalry as Revealed through Optokinetic Nystagmus, pp. 1–17 (1995)
- [13] Zhang, P., Engel, S., Rios, C., He, B., He, S.: Binocular rivalry requires visual attention: Evidence from EEG. *Journal of Vision* 9, 291 (2009)
- [14] Joordens, S.: When Timing the Mind One Should Also Mind the Timing: Biases in the Measurement of Voluntary Actions. *Consciousness and Cognition* 11, 231–240 (2002)
- [15] Lee, D.K., Koch, C., Braun, J.: Attentional capacity is undifferentiated: concurrent discrimination of form, color, and motion. *Percept. Psychophys.* 61, 1241–1255 (1999)
- [16] Pastukhov, A., Fischer, L., Braun, J.: Visual attention is a single, integrated resource. *Vision Research* 49, 1166–1173 (2009)
- [17] Orbach, J., Ehrlich, D., Heath, H.A.: Reversibility of the Necker cube. I. An examination of the concept of satiation of orientation. *Perceptual and Motor Skills* 17, 439–458 (1963)
- [18] Boynton, G.M.: Attention and visual perception. *Current Opinion in Neurobiology* 15, 465–469 (2005)
- [19] Reynolds, J.H., Chelazzi, L.: Attentional modulation of visual processing. *Annual Review of Neuroscience* 27, 611–647 (2004)
- [20] Serences, J.T., Yantis, S., Culberson, A., Awh, E.: Preparatory activity in visual cortex indexes distractor suppression during covert spatial orienting. *Journal of Neurophysiology* 92, 3538–3545 (2004)
- [21] Liu, L., Tyler, C.W., Schor, C.M.: Failure of rivalry at low contrast: evidence of a suprathreshold binocular summation process. *Vision Research* 32, 1471–1479 (1992)

Modelling Neurotic Psychopathology: Memory, Attention and Symbolization

Roseli S. Wedemann¹, Luís Alfredo V. de Carvalho², and Raul Donangelo³

¹ Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro
Rua São Francisco Xavier, 524, 20550-900, Rio de Janeiro, RJ, Brazil
roseli@ime.uerj.br

² Eng. Sistemas e Computação, COPPE - Universidade Federal do Rio de Janeiro,
Caixa Postal 68511, 21945-970, Rio de Janeiro, RJ, Brazil
LuisAlfredo@ufrj.br

³ Instituto de Física, Universidade Federal do Rio de Janeiro
Caixa Postal 68528, 21941-972, Rio de Janeiro, RJ, Brazil
donangel@if.ufrj.br

Abstract. In earlier work, we have proposed a neural network model that describes some mental processes involved in neuroses, by an associative memory mechanism, where modules corresponding to sensorial and symbolic memories interact, representing unconscious and conscious mental activity. Here, we relate our neuroses model with two models which have been proposed for modelling cognitive functions underlying consciousness: the CODAM model and the Dehaene-Kerszberg-Changeux Global Workspace model, to support that memory and attentional mechanisms are essential for consciousness.

1 Introduction

It is an important finding of psychoanalytic research regarding the *neuroses*, that traumatic and repressed memories are knowledge which is present in the subject but which is inaccessible to him, through symbolical representation [1]. It is momentarily or permanently inaccessible to the subject's conscience and is therefore considered *unconscious* knowledge. Freud observed that neurotic patients systematically repeated symptoms in the form of ideas and impulses and called this tendency a *compulsion to repeat*, which he related to repressed or traumatic memory traces [12]. Since the unconscious mind cannot be expressed symbolically, it does so through other body response mechanisms, in the form of neurotic (unconscious) symptoms, similar to reflexes.

By *symbolic expression* we refer to the association of symbols to meaning as in language and also other forms of expressing thought and emotions, such as artistic representations (e.g., a painting or musical composition) and remembrance of dreams. Neurotics have obtained relief and cure of painful symptoms through a psychoanalytic method called *working-through*, which aims at developing knowledge regarding the symptoms by accessing unconscious memories and understanding and changing the analysand's compulsion to repeat [2]. It involves mainly analyzing free associative talking, symptoms, parapraxes (slips of the tongue and pen, misreading, forgetting, etc.), dreams and also that which is acted out in transference.

We have described a model in [34], where we proposed that the neuroses manifest themselves as an associative memory process [5], where the network returns a stored pattern when it is shown another input pattern sufficiently similar to the stored one. We modelled the compulsion to repeat neurotic symptoms by supposing that such a symptom is acted when the subject is presented with a stimulus which resembles a repressed or traumatic memory trace. The stimulus causes a stabilization of the neural net onto a minimal energy state, corresponding to the memory trace that synthesizes the original repressed experience, which in turn generates a neurotic response (an *act*). The neurotic act is not a result of the stimulus as a new situation but a response to the repressed memory trace.

We mapped the linguistic, symbolic, associative process involved in psychoanalytic working-through into a corresponding process of reinforcing synapses among memory traces in the brain. These connections should involve declarative memory, leading to at least partial transformation of repressed memory to consciousness. This has a relation to the importance of language in psychoanalytic sessions and the idea that unconscious memories are those that cannot be expressed symbolically. The network topology which results from this reconfiguration process will stabilize onto new energy minima, associated with new acts.

2 Hierarchical Memory Model for the Neuroses

We proposed a memory organization [34], where neurons belong to two hierarchically structured modules corresponding to *sensorial* and *symbolic memories* (see Fig. 1). Traces stored in sensorial memory represent mental images of stimuli received by sensory receptors. Symbolic memory stores higher level representations of traces in sensorial memory, *i.e.* *symbols*, and represents brain structures associated with symbolic processing, language and consciousness. Sensorial and symbolic memories interact, producing unconscious and conscious mental activity. If the retrieval of a sensorial memory trace can activate retrieval of a pattern in symbolic memory, it can become conscious. We refer to the work of Edelman [6] for a neurophysiological discussion of these issues.

Brain neural topology is structured by cooperative and competitive mechanisms, controlled by neurosubstances, where neurons interact mainly with nearby neighbors, having fewer long-range synaptic connections to distant neurons [6]. This is started and controlled by environmental stimulation and is the process whereby the environment represents itself in the brain. We thus proposed a *clustering algorithm* based on these mechanisms [34] to model the self-organizing process which results in a hierarchical, clustered topology of each memory module. We represent the association of ideas or symbols (such as in culture and language) by long-range synapses in the model.

Memory functioning was originally modelled by a Boltzmann Machine (BM). However, the power-law and generalized q -exponential behavior we have found, for the node-degree distributions of the network topologies generated by our model, indicate that they may not be well described by Boltzmann-Gibbs (BG) statistical mechanics, but rather by Nonextensive Statistical Mechanics (NSM) [7]. We have thus modelled memory by a generalization of the BM called Generalized Simulated Annealing (GSA) [47], and this affects the chain of associations of ideas which we are modelling.

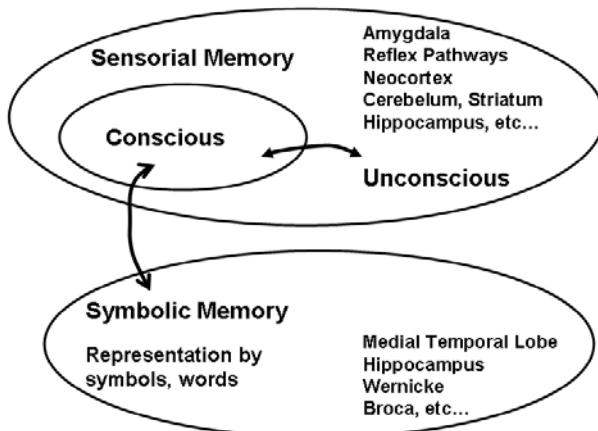


Fig. 1. Memory modules which represent storage of sensorial input and symbolic representations, and also memory traces which can or cannot become conscious

3 A Relation to Two Models for Consciousness

From the psychoanalytic description of neurotic processes we have mentioned above and our modelling experience, it seems that memory and attention are basic mechanisms which are important for the emergence of consciousness. In neurosis, a stimulus which, by an associative memory process [5], retrieves a repressed or traumatic memory trace sets goals for a resulting act. If the selected memory trace (which should be the attended object) can be expressed symbolically (reported) than it is considered that it can become conscious. If not, it will generate a neurotic unconscious act.

There are two models which seem to us, have similar mechanisms and which could enhance and complete our description of conscious and unconscious mental processes involved in neuroses. One is the CODAM model [8][9][10], where the “Working Memory for Report Module” could support simulation of the access from the conscious subset of the sensorial memory module of Fig. 1 to symbolic memory, allowing these traces to become conscious (available for report). Repressed or traumatic memories can be represented by inhibiting their connectivity to the working memory module. In some of Freud’s works, he referred to the conscious subset of Fig. 1 as the *preconscious* [11], *i.e.* that which may become conscious. It seems our associative memory retrieval mechanism could be used to implement the “Goal” and “Object Map” modules in CODAM, for generating the attention signal in neuroses.

These mechanisms are also present in the Dehaene-Kerszberg-Changeux Global Workspace (DKC-GW) model [12][13][14], which supports a goal driven signal for attentional amplification of an attended object for report. These authors propose a model for the simulation of effortful tasks on the basis of plausible molecular, anatomical, and functional features of the brain, which includes short and long-range synapses for a hierarchical, clustered organization of the network architecture. Such hierarchy and

clusterization is also basic for our neuroses model. Both Edelman [6] and Changeux [12][14] propose an organization of consciousness into levels, primarily on the basis of language use, which depends on the complexity of neural network organization.

The CODAM and DKC-GW models present common features in the attempt of modelling access to consciousness. These models propose that if a stimulus gains access to a specific neuronal subset, it becomes conscious. In CODAM this subset corresponds to the Working Memory (WM) module and in DKC-GW it is the Global Workspace (GW). Dehaene et al. [13] argue that the population of neurons that constitute GW do not belong to a distinct set of “cardinal” areas in the brain but is distributed among brain areas in variable proportions. Long-range axons connect distant cortical neurons in GW, to potentially interconnect multiple distributed and specialized brain areas in a coordinated, though variable manner. Stimuli that intensely mobilize access to WM or GW are available for operations such as thinking, reasoning, verbal report and symbolization, evaluation, memorization and action guidance, associated with the subjective feeling of conscious effort. Both models involve mechanisms for selecting (gating) a subset of input stimuli that will gain access to consciousness, functioning as an attentional filter, which amplifies the attended stimulus and inhibits distracters. Stimuli and memory traces which have inhibited access to WM and GW constitute the unconscious.

The models present error correction mechanisms for guaranteeing that the attended stimulus corresponds to specified goals. Mechanisms for sustaining the attended stimulus for a certain period of time in WM or GW are also included in both models.

Taylor [8][10] conjectures a special mechanism in CODAM called the “Corollary Discharge Module” which acts as a buffer of the attention copy signal for a short time. This module allows speed up of the access of the attended stimulus into WM and participates in an error correction scheme to inhibit distracters, providing greater efficiency in moving the focus of attention. The introduction of this corollary discharge mechanism allowed the simulation of results related to the attentional blink phenomenon [8] and, more recently, Taylor has argued that it is an essential mechanism for reproducing the sense of ownership and self in mental structure [10].

The findings of Freud regarding neurotic behavior and our neuroses model are in agreement with both the CODAM and DKC-GW models, in that unconscious memories have inhibited access to symbolic processing areas of the brain (this would correspond to inhibited access to WM or GW). We have proposed a mechanism whereby an input stimulus selects a stored long-term neurotic memory trace through an associative memory process. This would correspond to a goals module in CODAM. Our algorithm for simulating working-through describes a mechanism based on Hebbian learning, whereby repressed memory traces can gain access to symbolic processing areas in the brain. This could be integrated with the attentional amplification mechanisms of CODAM or DKC-GW, representing access of the repressed to WM and GW and thus to consciousness. Our algorithms involve distributed neuronal mechanisms, processed by a hierarchically clustered complex network, where long-range connections allow access to representations stored by clusters of neurons more tightly connected by short-range synapses.

Our main contribution in recent work has been to propose a neuronal model, based on known microscopical, biological brain mechanisms, that describes conscious and

unconscious memory activity involved in neurotic behavior, as described by Freud. We now relate this to the CODAM and the DKC-GW model to argue that memory and attentional mechanisms are essential for access to symbolic processing, language and meaning for consciousness. Our model is very schematic and we do not sustain or prove that this is the actual mechanism that occurs in the human brain. It nevertheless illustrates and seems to be a good metaphorical view of facets of mental phenomena, for which we seek a neuronal substratum, and suggests directions of search.

Acknowledgments. We acknowledge financial support from the Brazilian National Research Council (CNPq), the Rio de Janeiro State Research Foundation (FAPERJ) and the Brazilian agency which funds graduate studies (CAPES).

References

1. Freud, S.: *Introductory Lectures on Psycho-Analysis*. Standard edn. W. W. Norton and Company, New York (1966), First German edn (1917)
2. Freud, S.: *Beyond the Pleasure Principle*. Standard edn. The Hogarth Press, London (1974), First German edn. (1920)
3. Wedemann, R.S., Carvalho, L.A.V., Donangelo, R.: Network Properties of a Model for Conscious and Unconscious Mental Processes. *Neurocomputing* 71, 3367–3371 (2008)
4. Wedemann, R.S., Donangelo, R., Carvalho, L.A.V.: Generalized Memory Associativity in a Network Model for the Neuroses. *Chaos* 19, 015116(1–11) (2009)
5. Hertz, J.A., Krogh, A., Palmer, R.G. (eds.): *Introduction to the Theory of Neural Computation*. Lecture Notes, vol. I. Perseus Books, Cambridge (1991)
6. Edelman, G.M.: *Wider than the Sky, a Revolutionary View of Consciousness*. Penguin Books, London (2005)
7. Tsallis, C., Stariolo, D.A.: Generalized Simulated Annealing. *Physica A* 233, 395–406 (1996)
8. Fragapanagos, N., Kockelkoren, S., Taylor, J.G.: A Neurodynamic Model of the Attentional Blink. *Cognitive Brain Research* 24, 568–586 (2005)
9. Taylor, J.G.: CODAM Model: Through Attention to Consciousness. *Scholarpedia* 2(11), 1598 (2007)
10. Taylor, J.G.: A Neural Model of the Loss of Self in Schizophrenia. *Schizophrenia Bull*, sbq033V1–sbq033 (2010)
11. Freud, S.: *The Unconscious*. Standard edn, vol. XIV. The Hogarth Press, London (1957), First German edn (1915)
12. Changeux, J.P.: *L'homme de Vérité*. Editions Odile Jacob, Paris (2002)
13. Dehaene, S., Kerszberg, M., Changeux, J.P.: A Neuronal Model of a Global Workspace in Effortful Cognitive Tasks. *Proc. Natl. Acad. Sci. USA* 95, 114529–114534 (1998)
14. Changeux, J.P.: The Molecular Biology of Consciousness Investigated with Genetically Modified Mice. *Phil. Trans. R. Soc. B* 361, 2239–2259 (2006)

How to Use the SOINN Software: User's Guide (Version 1.0)

Kazuhiro Yamasaki¹, Naoya Makibuchi¹, Furao Shen², and Osamu Hasegawa¹

¹ Department of Computational Intelligence and Systems Science,

Tokyo Institute of Technology, Yokohama 226-8503, Japan

{yamasaki.k.ac,makibuchi.n.aa,hasegawa.o.aa}@titech.ac.jp
<http://www.haselab.info/>

² The State Key Laboratory for Novel Software Technology,

Nanjing University, China

frshen@nju.edu.cn

Abstract. The Self-Organizing Neural Network (SOINN) is an unsupervised classifier that is capable of online incremental learning. Studies have been performed not only for improving the SOINN, but also for applying it to various problems. Furthermore, using the SOINN, more intelligent functions are achieved, such as association, reasoning, and so on. In this paper, we show how to use the SOINN software and to apply it to the above problems.

Keywords: Self-Organizing Incremental Neural Network, SOINN software.

1 Introduction

The Self-Organizing Incremental Neural Network (SOINN) [1] is an online unsupervised mechanism proposed by Shen and Hasegawa which is capable of incremental learning, that is, it can learn new knowledge without destroying the old learned knowledge. Because the neurons in the network are self-organized, it is not necessary to define the network structure and size in advance. In addition, this system is robust to noise.

Studies have been performed not only for improving the SOINN, but also for applying it to various problems. For example, an Enhanced-SOINN (ESOINN) [2] has succeeded in reducing the number of parameters and layers of the original SOINN from two layers and eight parameters to one layer and four parameters. Furthermore, it is capable of separating clusters with a high-density overlap. An Adjusted SOINN Classifier (ASC) [3] automatically learns the number of prototypes needed to determine the decision boundary; thus, very rapid classification is achieved.

Using the SOINN, more intelligent functions can be achieved, such as association, reasoning, and so on. An associative memory (AM) system using the SOINN has been proposed [4], which is called the SOIAM. This system learns

a pair of vectors (a key vector and an associative vector), which makes it possible to realize the association between them. On the other hand, a novel AM system consisting of a three-layer architecture has also been proposed [5]. This system realizes the association of both static information and temporal sequence information. In addition, a pattern-based reasoning system using the SOINN has been also proposed [6], which achieves reasoning with the pattern-based if-then rules of propositional logic.

The SOINN is also applied to fields such as robotics (e.g., language acquisition [7][8], task planning [9][10], the SLAM system [11], and robot navigation [12]).

The SOINN software is available here¹. We provide an application and the source code written in the C++ language of the SOINN with a single layer and two parameters, which has been introduced by [3] and employed in [4][6]. In this paper, we show how to use this software, and then describe briefly how to extend the source code into one of [4] or [6].

2 Application

You can acquire the SOINN software as a solution file of Microsoft Visual Studio 2005. To run the SOINN application, open and build the “SOINN.sln” file, then execute the resulting program. In doing so, windows containing the menu, the Input, and the Output are displayed, as shown in Figure 1. The menu window provides the functions for the designation of the data set, the definition of parameters and noise, and so on. The Input window shows how the data set is input continuously, specifically online, as well as the current number of input data items. The Output window shows how the network grows in a self-organizational way with each input data item, and the current number of nodes and classes in the network.

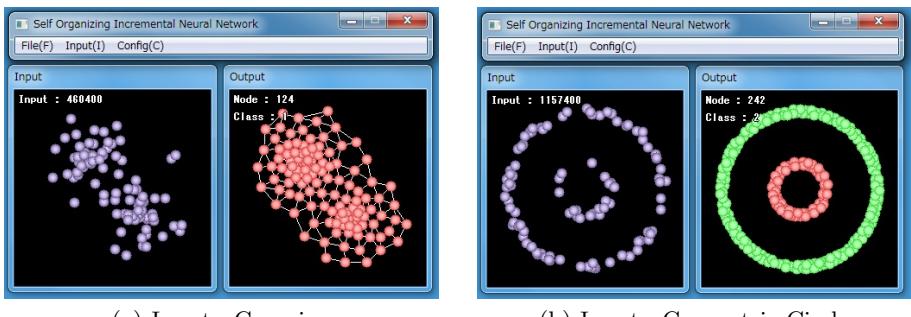
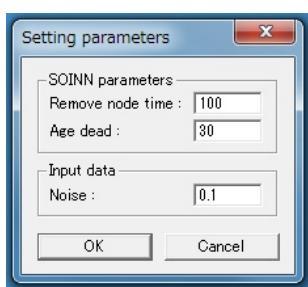
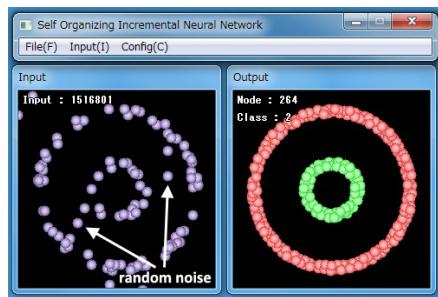


Fig. 1. Execution examples

If you select “Input > Synthetic data > Gaussian or Concentric circle,” data generated from the two gaussian or concentric circles is input into the network

¹ <http://www.haselab.info/soinn-e.html>

**Fig. 2.** Screen of Setting parameters**Fig. 3.** Data set including noise

over time. The SOINN displays the topological structure of the input data, so that it regards a set of all nodes connected with edges as a cluster. This makes it possible to classify unsupervised data. For example, Figure 1(b) shows the classified data of each circle using different colors. At the same time, the SOINN shows the probabilistic density of the input data. For example, in Figure 1(a), it is found that nodes on the area around the center of the gaussian are distributed in high density and vice versa. In addition, the SOINN realizes the compression of the input data by finding typical prototypes for a large-scale data set. In fact, in Figures 1(a) and 1(b), the numbers of input data are 460,400 and 1,157,400, and the numbers of output data are 124 and 242.

The parameters necessary for the SOINN can be set by selecting “Config > Setting parameters,” as shown in Figure 2. In this window, not only the SOINN parameters, but also the amount of noise added to the data set, can be defined. For example, as shown in Figure 2, if “Noise” is defined as 0.1, 10% random noise is added to the data set. However, as shown in Figure 3, it is found that the SOINN adequately displays the distribution of the original data set without being affected by noise.

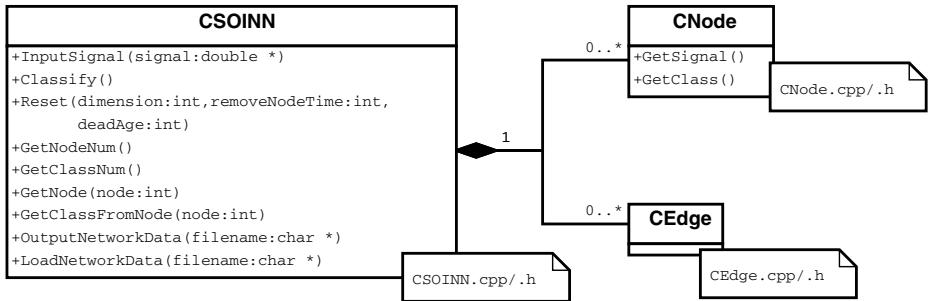
3 SOINN API

A brief class diagram is depicted in Figure 4. The SOINN API consists of six core files, `CSOINN.cpp/.h`, `CNode.cpp/.h`, and `CEdge.cpp/.h`. The class `CSOINN` represents the entire SOINN system. The `CNode` is the node (i.e., neuron) in the network. You can retrieve the information of the node using the API functions of this class. The `CEdge` is the edge between the two nodes.

Following are explanations of the SOINN API functions:

- `CSOINN::InputSignal()` represents the algorithm² of the SOINN for one given input data.
- `CSOINN::Classify()` assigns the class ID to all the nodes according to the current network structure.

² This function includes processes such as the addition or deletion of nodes.

**Fig. 4.** Class diagram of the SOINN software

- `CSOINN::Reset()` resets the parameters of the SOINN (i.e., dimension, `removeNodeTime`, and `deadAge`). Note that, in the current implementation, all the nodes and edges in the network are also removed.
- `CSOINN::GetNodeNum()` returns the number of nodes in the current network.
- `CSOINN::GetClassNum()` returns the number of classes in the current network.
- `CSOINN::GetNode()` returns the `CNode` instance with the same ID³ as the function’s argument.
- `CSOINN::GetClassFromNode()` returns the class ID of the `CNode` instance with the same ID as the function’s argument.
- `CSOINN::OutputNetworkData()` outputs the current network to a given file.
- `CSOINN::LoadNetworkData()` loads any network from a given file.
- `CNode::GetSignal()` returns the weight vector of the `CNode` instance itself.
- `CNode::GetClass()` returns the class ID of the `CNode` instance itself.

4 Usage Example

In this section, we show how to use the API of the SOINN software through the sample code (Listing 1) of the 1-Nearest Neighbor (1-NN) algorithm. Note that it is necessary to define the functions `Distance()` and `LoadSignal()`.

First, to use the SOINN, call the `CSOINN::CSOINN()` function. This constructor returns the `CSOINN` instance. Next, to perform the learning process of the SOINN, call the `CSOINN::InputSignal()` function. The argument of this function represents one training data vector. You then call the `CSOINN::Classify()` function to assign all nodes in the current network with the class ID.

Next, to run the 1-NN algorithm, calculate the distance between the test data and the weight vector of each of the nodes in the current network, where each weight vector is returned by the `CNode::GetSignal()` function. Finally, use the functions `CNode::GetClass()` or `CSOINN::GetClassFromNode()` to retrieve the class ID.

³ The SOINN software assigns an ID to all the nodes in the current network.

Listing 1. Sample code of the 1-NN using the SOINN API

```

1 // DIMENSION : the number of dimension of the input vector
2 // REMOVE_NODE_TIME : the predefined threshold
3 //           to remove edges with an old age
4 // DEAD_AGE : the predefined parameter
5 //           to delete nodes having only one neighbor
6 // TRAINING_DATA_FILE : the name of training data file
7 // TEST_DATA_FILE : the name of test data file
8 // *signal : the training data vector
9 // *targetSignal : the test data vector
10 //
11 // Distance() : the distance function (typically Euclidean distance)
12 // LoadSignal() : the function to get one training/test data vector
13
14 CSoInn *pSoInn;
15 pSoInn = new CSoInn( DIMENSION, REMOVE_NODE_TIME, DEAD_AGE );
16
17 double *signal;
18 while ( ( signal = LoadSignal( TRAINING_DATA_FILE ) ) != NULL ) {
19     pSoInn->InputSignal( signal );
20 }
21 pSoInn->Classify();
22
23 // 1-NN algorithm.
24 double minDist      = CSoInn::INFINITY;
25 int    nearestID    = CSoInn::NOT_FOUND;
26
27 double *targetSignal = LoadSignal( TEST_DATA_FILE );
28 for ( int i = 0; i < pSoInn->GetNodeNum(); i++ ) {
29
30     double *nodeSignal = pSoInn->GetNode( i )->GetSignal();
31     double dist        = Distance( targetSignal, nodeSignal );
32
33     if ( minDist > dist ) {
34         minDist      = dist;
35         nearestID   = i;
36     }
37
38 }
39
40 int    nearestClassID = pSoInn->GetNode( nearestID )->GetClass();
41 printf( "Nearest Node ID : %d, Class ID : %d.\n", nearestID, nearestClassID );
42
43 delete pSoInn;

```

5 Extensions

In this section, we briefly describe how to extend the SOINN to [2,3,4,5,6]. In the SOIAM [4], the dimension of the association pair must be defined in CSoInn and CNode. Furthermore, it is necessary to define an appropriate distance function, a recall function, and so on. Listing 2 shows the sample code of the additional implementation for the SOIAM. In the other AM system consisting of a three-layer architecture [5], the new class representing an input layer, a memory layer, and an associative layer must be defined. In a pattern-based reasoning system [6], as well as in the SOIAM, some functions, such as a distance function, also need to be defined. In the ESOINN [2], it is necessary to modify the CSoInn::InputSignal() function to enable this system to separate clusters with high-density overlap. In addition, two new parameters to determine the noise

node must be added in CSOINN. In the ASC [3], the multiple CSOINN instances that correspond to the classes of the data set are required, and the new class to perform the k -means algorithm for the stabilization of the ASC must be defined. The functions to reduce noisy and unnecessary prototype nodes must also be defined.

Listing 2. Sample code of the additional implementation of the SOIAM

```

1  class CSOIAM : public CSOINN {
2
3  public :
4      std::vector<CNode *> *Recall( double *signal, bool isDirect );
5
6  private :
7      int keyDim;
8      int assocDim;
9
10     double Distance( double *signal1, double *signal2, bool isDirect );
11
12 }
13 /* ===== */
14 std::vector<CNode *> *CSOIAM::Recall( double *signal, bool isDirect )
15 {
16     int classNum = GetClassNum();
17     int nodeNum = GetNodeNum();
18
19     std::vector<std::pair<double, int>> minPair( classNum );
20     for ( int i = 0; i < nodeNum; i++ ) {
21
22         CNode *node = GetNode( i );
23         int classID = node->GetClass();
24
25         double distance = Distance( signal, node->GetSignal(), isDirect );
26         if ( minPair[classID].first > distance ) {
27             minPair[classID].first = distance;
28             minPair[classID].second = i;
29         }
30     }
31
32
33     std::vector<CNode *> *recalledNodes = new std::vector<CNode *>( classNum );
34     for ( int i = 0; i < classNum; i++ ) {
35         recalledNodes->at( i ) = GetNode( minPair[i].second );
36     }
37
38     return recalledNodes;
39
40 }
41 /* ===== */
42 double Distance( double *signal1, double *signal2, bool isDirect )
43 {
44     int beginPtr = ( isDirect ) ? 0 : keyDime;
45     int endPtr = ( isDirect ) ? keyDim : ( keyDim + assocDim );
46     int normConst = ( isDirect ) ? keyDim : assocDim;
47
48     double distance = 0.0;
49     for ( int i = beginPtr; i < endPtr; i++ ) {
50         double tmp = ( signal1[i] - signal2[i] );
51         distance += ( tmp * tmp );
52     }
53     distance = sqrt( distance / normConst );
54
55     return distance;
56
57 }
```

Acknowledgement

This study was supported by the Industrial Technology Research Grant Program in 2010 from the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

1. Shen, F., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. *Neural Networks* 19(1), 90–106 (2005)
2. Shen, F., Ogura, T., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks* 20(8), 893–903 (2007)
3. Shen, F., Hasegawa, O.: A fast nearest neighbor classifier based on self-organizing incremental neural network. *Neural Networks* 21(10), 1537–1547 (2008)
4. Sudo, A., Sato, A., Hasegawa, O.: Associative memory for online learning in noisy environments using self-organizing incremental neural network. *IEEE Trans. on Neural Networks* 20(6), 964–972 (2009)
5. Shen, F., Yu, H., Kasai, W., Hasegawa, O.: An Associative Memory System for Incremental Learning and Temporal Sequence. In: Proc. of the 2010 International Joint Conference on Neural Networks (to appear, 2010)
6. Shen, F., Sudo, A., Hasegawa, O.: An online incremental learning pattern-based reasoning system. *Neural Networks* 23(1), 135–143 (2010)
7. He, X., Kojima, R., Hasegawa, O.: Developmental Word Grounding through A Growing Neural Network with A Humanoid Robot. *IEEE Trans. SMC-Part B* 37(2), 451–462 (2007)
8. He, X., Ogura, T., Satou, A., Hasegawa, O.: Developmental Word Acquisition And Grammar Learning by Humanoid Robots through A Self-Organizing Incremental Neural Network. *IEEE Trans. SMC-Part B* 37(5), 1357–1372 (2007)
9. Makibuchi, N., Shen, F., Hasegawa, O.: Online Knowledge Acquisition and General Problem Solving in a Real World by Humanoid Robots. In: 1st SOINN Workshop, in Conjunction with ICANN (2010)
10. Makibuchi, N., Shen, F., Hasegawa, O.: General Problem Solving System in a Real World and Its Implementation on a Humanoid Robot. *IEICE Trans. Inf. & Sys.* J93-D(6), 960–977 (2010) (in Japanese)
11. Kawewong, A., Honda, Y., Tsuboyama, M., Hasegawa, O.: Reasoning on the Self-Organizing Incremental Associative Memory for Online Robot Path Planning. *IEICE Trans. Inf. & Sys.* E93-D(3), 569–582 (2010)
12. Tangruamsub, S., Tsuboyama, M., Kawewong, A., Hasegawa, O.: Unguided Robot Navigation using Continuous Action Space. In: 1st SOINN Workshop, in Conjunction with ICANN (2010)

Unguided Robot Navigation Using Continuous Action Space

Sirinart Tangruamsub, Manabu Tsuboyama,
Aram Kawewong, and Osamu Hasegawa

Department of Computational Intelligence and Systems Science,
Tokyo Institute of Technology

(tangruamsub.s.aa, kawewong.a.aa, hasegawa.o.aa)@m.titech.ac.jp,
m_tuboyama@hotmail.co.jp

Abstract. In this paper, we propose a new method for robot vision-based navigation. This method is distinct from other methods, in the sense that it works even when the actions are not pre-determined and when their space is continuous. An if-then rule is introduced in the form of (from-position \wedge action \rightarrow to-position) for path and action planning. These if-then rules are learned incrementally in an unsupervised manner. When planning the path, the sequence of path fragments is retrieved from these rules. An experiment is performed with a realistic robot simulator called Webots; the results show that our method succeed in reaching our goals.

Keywords: Robot Navigation, Incremental Learning, Self-Organising Incremental Neuron Network (SOINN).

1 Introduction

Vision-based robot navigation is an interesting topic in the field of robotics. Its task is to localise the position of a mobile robot and to help a robot find the path to a goal.

One of the objectives of vision-based navigation is that the robot should be able to localise itself despite having an incomplete map; this engenders a well-known Simultaneous Localisation and Mapping (SLAM) problem. To date, many works have been undertaken to solve the SLAM problem. However, most of these works ignore path planning, and merely assume that the robot uses guided exploration (i.e. it is controlled by humans to collect data); thus, the problem solely focuses on SLAM. Controlling the robot to move from the current position to a neighbouring place (or to a node in the topological map) can be done by hard-coding the action or by a visual servoing technique.

Another method for robot navigation is reinforcement learning [1][2]. This method views the world as state transitions composed of an initial state, an action state and a final state. In this method, the robot needs no human guidance. However, to limit the search space, the actions of the robot are pre-determined.

Another problem in vision-based navigation for mobile robots is that many proposed methods are not incremental. For a specific limited environment in which the

spaces are few the number of nodes in a topological map can be designed properly, which produces a highly accurate map [3][4]. The main disadvantage of these methods is that they are unsuitable for application in large-scale environments. Some researchers have proposed incremental map building methods. Valgren and Lilienthal [5][6] proposed incremental spectral clustering (ISC), which is adapted from a spectral clustering algorithm. However, this method has the following drawbacks: the number of nodes always increases because of iteration, large amounts of time and memory are required to generate the affinity matrix and the image-classification accuracy tends to drop when images containing noise are tested.

As described herein, we propose a method for robot exploration with no prior knowledge. The robot action is *continuous* and it moves in unlimited directions. We propose the use of the associative memory implemented on Self-Organising Incremental Neural Networks (SOINNs), as proposed by Shen et al. [7]. The advantage of this method is that the robots need not memorise collected images, whereas other methods, such as visual servoing, must record all the input data for comparison. The output of this method is the topological map, the nodes of which are linked by self-generated actions. Unlike ISC, the number of nodes can be adjusted in the topological map, depending on the time and environment.

2 System Overview

We propose a method for finding a path for a robot by discovering a sequence of path fragments and making the robot follow that sequence until it reaches a goal. We define if-then rules to search for such path fragments. These rules consist of two parts: *conditional* and *sequential*. The conditional part includes the from-position or the current position, q and a robot action, a . The sequential part shows a to-position, f when the conditional part is accomplished. This can be shown as

$$q \wedge a \rightarrow f$$

In this work, we use the image features representing the position data (q and f) and the robot motor parameter demonstrating the robot action data (a).

2.1 Learning System

In this section, the if-then rule learning is described. The learning method is based on the SOINN, as proposed by Shen et al. [7]. The SOINN is an online, unsupervised learning mechanism that determines cluster boundaries on the basis of the unlabeled data. A two-layer neural network is adapted for the SOINN structure. This model needs neither prior knowledge nor predefined network structure to learn. The network structure is adapted while learning by eliminating the unnecessary nodes and by uniting nodes when the number of nodes increases. This property makes the SOINN tolerant to noise. Moreover, the SOINN is designed for pattern data which has continuous space. This method is suitable for real-world robot applications for which the input data is not limited and unpredictable. Because the SOINN is an incremental learning machine, using it enables the robot to combine new knowledge with learned knowledge.

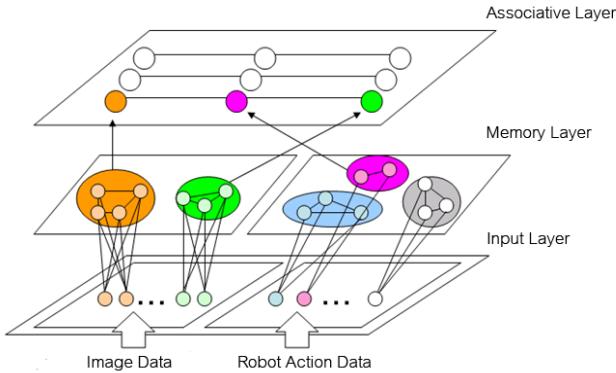


Fig. 1. Learning system architecture

To begin the learning process, a robot collects data. This data is sent to the learning system to learn the association among the data. This association can be viewed as if-shown in Fig. 1. The first layer is the input layer, whose role is to receive data collected by the robot. Because there are two types of data, action data and image data, we separate the input layer into two parts to support the input from the two data types. The second layer is the memory layer. The role of this layer is to memorise the data pattern from the first layer. The memory in this layer can be divided into temporary memory, short-term memory and long-term memory. When a constant number of input data frequency is reached, this input will shift from temporary memory to short-term memory. Data is shifted from the short-term memory to the long-term memory when the error margin of the intermediate data storage and the short-term memory are smaller than that of a constant value. The SOINN is used in the short-term and long-term memory for memorising data.

The third layer is an *associative* layer. This layer shows the relationship between the image data and the action data in the memory layer. This association can be viewed as if-then rule; each rule is represented as an association of three elements: (position element)-> (robot action element)->(position element).

The learning phase begins by allowing the robot to walk around an unknown place in arbitrary directions. The images and robot action information (motor parameters) are collected gradually. For each movement, the image feature of the current position is extracted as *from-position* data, and the motor parameters are collected as robot *action* data. After performing an action and reaching to the next position, the image feature of the *to-position* is extracted. Then, these image features of a from-position, a to-position and an action data (motor parameter) are submitted sequentially to the input layer. These inputs are sent to the memory layer for memorising, and then the association in the associative layer is constructed.

2.2 Map Building

A map can be built from the association in the associative layer. This map can be viewed as a topological map in which a single node represents a single element in the

association. The map building procedure begins when one if-then association (complete $q \wedge a \rightarrow f$) exists in the associative layer. The building is done by creating two nodes for the from-position element q and the to-position element f , and then connecting them by an associated action. After this initialisation, when a new association is created in the associative layer, the existence of the from-position element node and the to-position element node is checked. New nodes are inserted if the map contains none of these nodes. An edge is created to connect these two nodes.

2.3 Path Planning

To recognise the path to the assigned goal position, the robot first localises itself to determine the location on a map. This can be done by finding the nearest node of the input position in the memory layer. We set a distance threshold σ to detect situations where the robot is lost. The system rejects the input if the distance between the image input and all nodes in the memory layer exceeds this threshold. After recognising the node for both the start and goal positions, the shortest path algorithm is applied on a map. In this work, we use the Dijkstra algorithm [9]. The action that a robot should take for moving from the current node to the next node can be considered from the action part a (the robot action in the conditional part) of the if-then rules in the associative layer. While walking to the goal, the robot must localise itself by checking its current position on a map. When the distance between the input image feature and the nodes in the memory layer is compared, the predefined distance threshold σ is considered. The system rejects the input if the distance exceeds this threshold.

3 Experiment

3.1 Experimental Setup

To show that our proposed method is useful for robot navigation, we simulated the environments for robot exploitation. The experiment was performed using the Webot robot simulator [8], which is a realistic 3D physical simulator from which the system can be transferred easily to a real robot. To make the simulation world realistic, we collected images from a real-world environment and used these images as textures of objects in the simulated environment. The images were taken at the Suzukakedai Campus. The simulated environment is shown in Fig. 2.

In the first stage, we assume that the robot has no information about the simulation space. The robot must explore the space randomly with the continuous action space and without any human guidance, and then create a map and construct if-then rules. The images are taken and the robot actions are recorded during exploration and are then submitted to the input layer, as explained in Sect. 2.1. The map and if-then rules are then constructed simultaneously while the robot is exploring. After completing the learning phase, we test the correctness of our learned map and if-then rules by defining a start and goal position and allowing the robot to find a path between them.

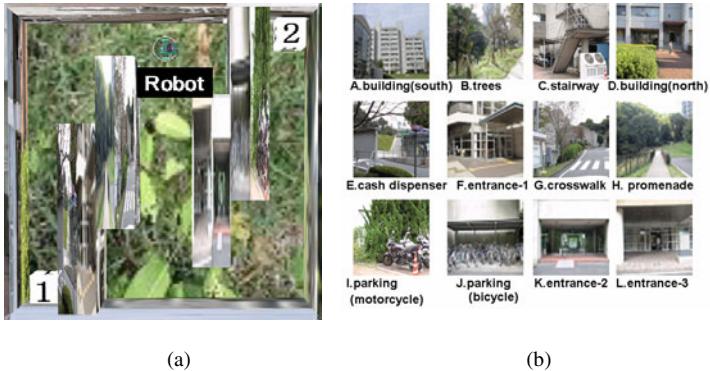


Fig. 2. Experimental environment: (a) simulated location created by Webot and (b) image used for map simulation in Webot

3.2 Experiment Setup

We divided the input data into two groups: images and robot actions. We do not use GPS to detect the robot position. Therefore, the images are used for position information. For the training phase, the images are collected by an e-puck robot simulated by the Webot program. These images are captured as grey-scale images with 40×40 resolution. Subsequently, each image is also captured with a subpart of itself. This can be done by setting a sliding window with 24×24 resolution, and moving it in an arbitrary direction throughout an original image for a fixed number of times. The sub-image inside this window will be captured for each sliding. For the test phase, the robot performs the process in the same way as was used for obtaining the training images but capturing only one sub-image from this window at the centre of image.

To represent images, we simply use the image pixel as a feature vector. The feature vectors of the sub-images are represented using 576 dimension vectors. These vectors are then used as the input feature in the input layer.

In addition, robot action data are used for creating if-then rules. In this work, we define this value as a tuple of the degree of robot wheels: left wheel and right wheel (L, R). These values are continuously set at 0–1.

3.3 Experimental Results

After exploring the simulated space, we allow the robot to find a path from a predefined start position to a goal position. Because the SOINN combines similar images into the same group, the network size in the memory layer is reduced.

We chose the first image taken by the e-puck robot in the training phase as the start position, and the 40th and 100th images as the goal positions. The results are shown in Fig. 3(a), where the 40th image of the training step is the goal, and Fig. 3(b), which shows the 100th image as the goal. In Fig. 3(a), the robot begins from the start position and then takes an action $(L, R) = (-1.00, 1.00)$ until it reaches the goal. In Fig. 3(b), the robot begins from the same place as in Fig. 3(a), and then takes an action sequence $(L, R) = (-1.00, 1.00) \rightarrow (1.00, 0.998) \rightarrow \dots \rightarrow (1.00, 0.998)$ before it reaches the goal.

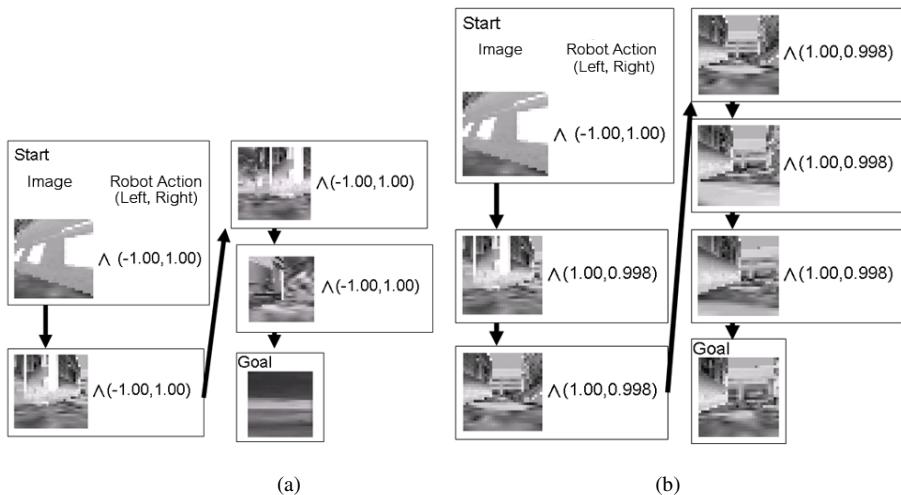


Fig. 3. Action sequences of robot: the defined goals are (a) 40th trained image and (b) 100th trained image

The results of the experiments show that our method is useful for robot navigation even when there is no prior knowledge. Because the learning system is based on the SOINN, which does not need to memorise all the input data, the memory usage for robot navigation is reduced, especially when the space is large. Moreover, the SOINN is designed for incremental learning with no pre-knowledge; the robot can integrate the new knowledge it receives with its previous knowledge.

Using neither hard-coding nor visual servoing, the robot can retrieve its next action from the appropriate association. This association is useful for deriving a topological map, which is then used for finding a sequence of path fragments, together with its corresponding sequence of actions, to reach the destination.

Acknowledgement

This study was supported by the Industrial Technology Research Grant Program in 2004 from New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

1. Smart William, D., Pack Kaelbling, L.: Effective Reinforcement Learning for Mobile Robots. In: International Conference on Robotics and Automation (2002)
2. Wang, Y., Cook, D.J., et al.: User-Guided Reinforcement learning of Robot Assistive Tasks for an Intelligent Environment. IEEE Trans. Intelligent Robots and System 1, 424–429 (2003)
3. Se, S., et al.: Vision-Based Global Localization and Mapping for Mobile Robots. IEEE Trans. Robotics 21(3), 364–375 (2005)

4. Grudic, G., Mulligan, J.: Topological Mapping with Multiple Visual Manifolds. In: Proc. Robotics: Sciences and Systems (2005)
5. Valgren, C., et al.: Incremental Spectral Clustering and Its Application To Topological Mapping. In: Proc. IEEE Int. Conf. Robotics and Automation (2007)
6. Valgren, C., Lilienthal, A.: Incremental Spectral Clustering and Seasons: Appearance-Based Localization in Outdoor Environments. In: IEEE Int. Conf. Robotics and Automation (2008)
7. Shen, F., Sudo, A., Hasegawa, O.: An online incremental learning pattern-based reasoning system. *Neural Networks* 23(1), 135–143 (2010)
8. Michel, O.: WebotsTM: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems* 1(1), 39–42 (2004)
9. Dijkstra, E.W.: A note on two problems in connexion with graphs. *Numerische Mathematik*, 269–271 (1959)

Self-Organizing Incremental Neural Network and Its Application

Furao Shen^{1,2} and Osamu Hasegawa³

¹ National Key Laboratory for Novel Software Technology, Nanjing University, China
frshen@nju.edu.cn

<http://cs.nju.edu.cn/rinc/>

² Jiangyin Information Technology Research Institute, Nanjing University, China

³ Imaging Science and Engineering Lab., Tokyo Institute of Technology, Japan
hasegawa@isl.titech.ac.jp

Abstract. Self-organizing incremental neural network (SOINN) is introduced. SOINN is able to represent the topology structure of input data, incrementally learn new knowledge without destroy of learned knowledge, and process online non-stationary data. It is free of prior conditions such as a suitable network structure or network size, and it is also robust to noise. SOINN has been adapted for unsupervised learning, supervised learning, semi-supervised learning, and active learning tasks. Also, SOINN is used for some applications such as associative memory, pattern-based reasoning, word-grounding, gesture recognition, and robotics.

Keywords: Self-organizing incremental neural network; Incremental learning; Competitive learning.

1 Introduction

Incremental learning addresses the ability of repeatedly training a network using new data without destroying the old prototype patterns. The fundamental issue for incremental learning is how a learning system can adapt to new information without corrupting or forgetting previously learned information: the so-called Stability-Plasticity Dilemma [1].

Numerous online incremental learning algorithms based on competitive neural networks have been proposed and applied in many applications. A salient advantage of competitive neural networks is their capability of operating with information of new data incrementally. Well known examples of competitive neural networks include Kohonen's self-organizing map (SOM) and its modification for supervised learning, Learning Vector Quantization (LVQ) [2]. SOM and LVQ are unsuitable for incremental learning, the number of nodes of such methods is predefined. Self-growing type SOM [3] can improve their performance by insertion of nodes and the gradual increase of their structural complexity. In addition, the Growing Neural Gas (GNG) architecture [4] is a modification to the GCS, in which the dimensionality of topological structures is not predefined, but is instead discovered during training. In self-growing SOM, GCS, and GNG, the

number of nodes is not predefined and, by insertion of nodes, new information can be learned. However, these methods require predefinition of the maximum number of nodes because insertion of nodes continues interminably as long as new input patterns come. In this regard, if the number of nodes reaches the maximum, further inputs engender the possibility of destruction.

In this paper, we introduce SOINN [5] for online incremental learning. In SOINN, the insertion of nodes is stopped and restarted automatically with new input patterns. Thereby, it avoids the indefinite increase of nodes, and it intends to present a balance of stability and plasticity [5].

2 Self-Organizing Incremental Neural Network (SOINN)

A SOINN adopts a two-layer network. The first layer learns the density distribution of the input data and uses nodes and edges to represent the distribution. The second layer separates clusters by detecting the low-density area of input data, and uses fewer nodes than the first-layer to represent the topological structure of input data. When the second-layer learning is finished, SOINN reports the number of clusters and gives typical prototype nodes of every cluster. It also adopts the same learning algorithm for the first and second layers. Fig. 1 shows a flowchart of the SOINN learning process.

When an input vector is given to SOINN, it finds the nearest node (winner) and the second nearest node (runner up) of the input vector. It subsequently judges if the input vector belongs to the same cluster of the winner or runner up using the similarity threshold criterion. The first layer of SOINN adaptively updates the similarity threshold of every node because the input data distribution is unknown. If node i has neighbor nodes, the similarity threshold T_i is calculated using the maximum distance between node i and its neighboring nodes.

$$T_i = \max_{j \in N_i} \|\mathbf{W}_i - \mathbf{W}_j\| \quad (1)$$

Therein, N_i is the set of neighbor nodes of node i and \mathbf{W}_i is the weight vector of node i . A similarity threshold T_i is defined as the minimum distance between node i and other nodes in the network if node i has no neighbor nodes.

$$T_i = \min_{j \in N \setminus \{i\}} \|\mathbf{W}_i - \mathbf{W}_j\| \quad (2)$$

Here, N is the set of all nodes.

The input vector will be inserted to the network as a new node to represent the first node of a new class if the distance between the input vector and the winner or runner up is greater than the similarity threshold of a winner or runner up. This insertion is called a between-class insertion because this insertion will engender the generation of a new class, even if the generated new class might be classified to some older class in the future.

If the input vector is judged as belonging to the same cluster of winner or second winner, and if no edge connects the winner and second winner, connect

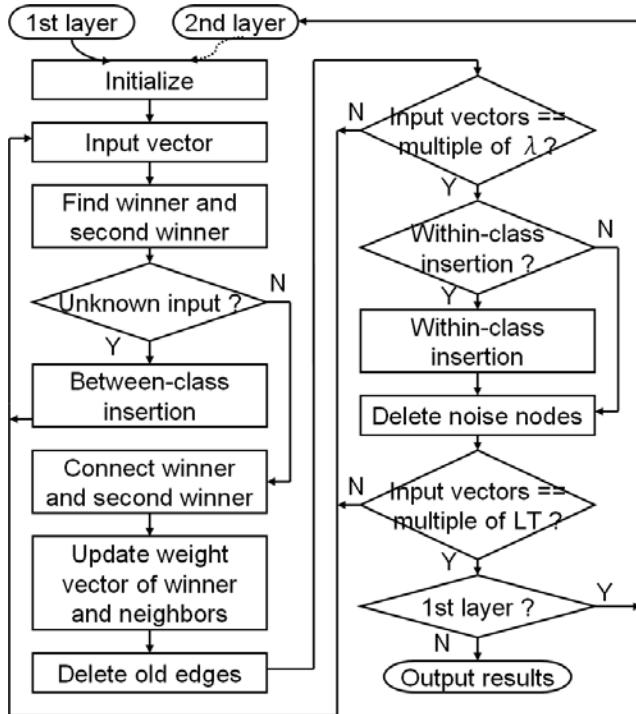


Fig. 1. Flowchart of SOINN

the winner and second winner with an edge, and set the ‘age’ of the edge as ‘0’; subsequently, increase the age of all edges linked to the winner by ‘1’.

Then, update the weight vector of the winner and its neighboring nodes. We use i to mark the winner node, and M_i to show the times for node i to be a winner. The change to the weight of winner $\Delta\mathbf{W}_i$ and change to the weight of the neighbor node $j (\in N_i)$ of i $\Delta\mathbf{W}_j$ are defined as $\Delta\mathbf{W}_i = \frac{1}{M_i}(\mathbf{W}_s - \mathbf{W}_i)$ and $\Delta\mathbf{W}_j = \frac{1}{100M_i}(\mathbf{W}_s - \mathbf{W}_j)$, where \mathbf{W}_s is the weight of the input vector.

Then, SOINN removes the edges whose age is larger than the value of threshold-old parameter age_{max} .

After λ learning iterations, the SOINN inserts new nodes into the position where the accumulating error is extremely large. Cancel the insertion if the insertion cannot decrease the error. The insertion here is called within-class insertion because the new inserted node is within the existing class; also, no new class will be generated during the insertion.

To delete the nodes created by noise, SOINN uses the following strategy: if the number of input signals generated so far is an integer multiple of a parameter λ , remove those nodes that have no neighbor or only one topological neighbor.

After LT learning iterations of the first layer, the learning results are used as the input for the second layer. The second layer of SOINN uses the same

learning algorithm as the first layer. For the second layer, the similarity threshold is constant; it is calculated using the within-cluster distance and between-cluster distance [5]. With a large constant similarity threshold, different from that of the first layer, the accumulation error for nodes of the second layer will be very high, and within-class insertion plays a major role in the learning process. With a large constant similarity threshold, the second layer also can delete some “noise nodes” that remain undeleted during first-layer learning.

3 Applications with SOINN

After the publishing of original SOINN, we adopted SOINN for some machine learning tasks and some applications.

1. Unsupervised learning

The original SOINN [5] and its enhanced version [6] are about unsupervised learning. SOINN is used to learn the topology structure of input data, it is able to grow incrementally and to accommodate input patterns of online non-stationary data distribution. It can separate classes with low-density overlap and detect the main structure of clusters that are polluted by noise. It automatically learns number of nodes and structure of the network, reports the number of clusters, and give the typical prototypes of every cluster.

2. Supervised learning

In [7], a prototype-based classifier based on SOINN is proposed. We firstly use SOINN on every class separately and generate typical prototypes for each class; then we do k -means clustering to tune the results of SOINN; then we adopt a k -Edited Neighbors Classifier like technique to reduce the noise influence of input data; at last, the input data of all classes and the results of noise-reduction part are used to clean the central part of every class and only boundary prototypes are remained. This method automatically learns the number of prototypes needed to determine the decision boundary. For different classes, the learned prototypes may be different. It is robust to noisy training data, and it realized very fast classification.

3. Semi-supervised learning

In [8], an online incremental semi-supervised learning method based on SOINN is presented. Using labeled data and a large amount of unlabeled data, the proposed method automatically learns the topology of input data distribution with no prior knowledge, it subsequently labels all generated nodes and divide the learned topology structure into sub-structures corresponding to classes. Weights of nodes are used as prototype vectors to realize classification. During the learning, new labeled or unlabeled data is able to incrementally add to the system.

4. Active learning

In [9], an online incremental active learning algorithm based on SOINN is proposed. It uses SOINN to represent the topology structure of input data, and then separates the generated nodes into different groups and subclusters. It then actively labels some teacher nodes and uses such teacher nodes

to label all unlabeled nodes. It queries the labels of some important samples rather than selecting the labeled samples randomly. It automatically learns the number of nodes and teacher vectors required for a current task. Experiments using artificial data and real-world data show that the proposed method works effectively and efficiently.

5. Associative memory

Associative memory operating in a real environment must perform well in online incremental learning and be robust to noisy data because noisy associative patterns are presented sequentially in a real environment. In [10], an associative memory is proposed to satisfy these requirements. New associative pairs that are presented sequentially can be learned accurately without forgetting previously learned patterns. The memory size of the proposed method increases adaptively with learning patterns. Therefore, it suffers neither redundancy nor insufficiency of memory size, even in an environment in which the maximum number of associative pairs to be presented is unknown before learning. The proposed associative memory performs as a bidirectional one-to-many or many-to-one associative memory and deals not only with bipolar data, but also with real-valued data.

6. Pattern-based reasoning

In [11], an architecture for reasoning with pattern-based if-then rules is proposed. By processing patterns as real-valued vectors and classifying similar if-then rules into clusters in the long-term memory, the proposed system can store pattern-based if-then rules of propositional logic, including conjunctions, disjunctions, and negations. It also achieves some important properties for intelligent systems such as incremental learning, generalization, avoidance of duplicate results, and robustness to noise. Experiments show that the proposed method is very effective for intelligent systems solving varying tasks autonomously in a real environment.

7. Other applications

SOINN is also used for humanoid robots [12], online robot navigation in the real-world [13], word acquisition and grammar learning [14], word grounding [15], and so on. We will not give the detailed discussion here.

4 Conclusion

In this paper, we introduced self-organizing incremental neural network and its application. There are still lots of problems to be solved. Such problems include how to build a theoretical basis of SOINN and how to use SOINN as a data- and feature-extraction method for the solving of large-scale problems. In addition, we need to discuss how SOINN can be used for different types of incremental learning, such as example-incremental, class-incremental, and attribution-incremental learning.

Acknowledgement

This work was supported in part by the Fund of the National Natural Science Foundation of China (Grant No. 60975047, 60723003, 60721002) and Jiangsu NSF grant (#BK2009080).

References

1. Grossberg, S.: Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks* 1, 17–61 (1988)
2. Kohonen, T. (ed.): *Self-Organizing Maps*. Springer Series in Information Sciences. Springer, Heidelberg (1997)
3. Bauer, H., Villmann, T.: Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks* 8(2), 218–226 (1997)
4. Fritzke, B.: A Growing Neural Gas Network Learns Topologies. In: *Neural Information Processing Systems*, vol. 7, pp. 625–632. MIT Press, Denver (1995)
5. Shen, F., Hasegawa, O.: An Incremental Network for On-line Unsupervised Classification and Topology Learning. *Neural Networks* 19(1), 90–106 (2006)
6. Shen, F., Ogura, T., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks* 20, 893–903 (2007)
7. Shen, F., Hasegawa, O.: A Fast Nearest Neighbor Classifier Based on Self-organizing Incremental Neural Network. *Neural Networks* 21, 1537–1547 (2008)
8. Kamiya, Y., Ishii, T., Shen, F., Hasegawa, O.: An Online Semi-Supervised Clustering Algorithm Based on a Self-organizing Incremental Neural Network. In: *The 2007 International Joint Conference on Neural Networks (IJCNN 2007)*, Orlando, FL, USA (August 2007)
9. Shen, F., Sakurai, K., Kamiya, Y., Hasegawa, O.: An Online Semi-supervised Active Learning Algorithm with Self-organizing Incremental Neural Network. In: *The 2007 International Joint Conference on Neural Network (IJCNN 2007)*, Orlando, FL, USA (August 2007)
10. Sudo, A., Sato, A., Hasegawa, O.: Associative Memory for Online Learning in Noisy Environments Using Self-organizing Incremental Neural Network. *IEEE Transactions on Neural Networks* 20(6), 964–972 (2009)
11. Shen, F., Sudo, A., Hasegawa, O.: An online incremental learning pattern-based reasoning system. *Neural Networks* 23, 135–143 (2010)
12. Okada, S., Kobayashi, Y., Ishibashi, S., Nishida, T.: Incremental learning of gestures for human-robot interaction. *AI & Society Journal of Knowledge, Culture and Communication* (2009)
13. Kawewong, A., Honda, Y., Tsuboyama, M., Hasegawa, O.: Reasoning on the Self-Organizing Incremental Associative Memory for Online Robot Path Planning. *IEICE Transactions on Information and Systems* E93-D(3), 569–582 (2010)
14. He, X., Ogura, T., Satou, A., Hasegawa, O.: Developmental Word Acquisition And Grammar Learning by Humanoid Robots through A Self-Organizing Incremental Neural Network. *IEEE Trans. SMC-Part B* 37, 1357–1372 (2007)
15. He, X., Kojima, R., Hasegawa, O.: Developmental Word Grounding through A Growing Neural Network with A Humanoid Robot. *IEEE Trans. SMC-Part B* 37, 451–462 (2007)

Machine Learning Approaches for Time-Series Data Based on Self-Organizing Incremental Neural Network

Shogo Okada¹, Osamu Hasegawa², and Toyoaki Nishida¹

¹ Dept. of Intelligence Science and Technology, Graduate School of Informatics,
Kyoto University

Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 Japan

{okada_s,nishida}@i.kyoto-u.ac.jp

² Imaging Science and Engineering Laboratory, Tokyo Institute of Technology
4259 Nagatsuta,Midori-ku, Yokohama, 226-8503 Japan

hasegawa@isl.titech.ac.jp

Abstract. In this paper, we introduce machine learning algorithms of time-series data based on Self-organizing Incremental Neural Network (SOINN). SOINN is known as a powerful tool for incremental unsupervised clustering. Using a similarity threshold based and a local error-based insertion criterion, it is able to grow incrementally and to accommodate input patterns of on-line non-stationary data distribution. These advantages of SOINN are available for modeling of time-series data. Firstly, we explain an on-line supervised learning approach, SOINN-DTW, for recognition of time-series data that are based on Dynamic Time Warping (DTW). Second, we explain an incremental clustering approach, Hidden-Markov-Model Based SOINN (HBSOINN), for time-series data. This paper summarizes SOINN based time-series modeling approaches (SOINN-DTW, HBSOINN) and the advantage of SOINN-based time-series modeling approaches compared to traditional approaches such as HMM.

1 Introduction

SOINN^[1] is known as a powerful tool for incremental unsupervised clustering. Using a similarity threshold and a local error-based insertion criterion, it is able to grow incrementally and to accommodate input patterns of on-line non-stationary data distribution. Using SOINN's function, which represents a distribution of input data and report the reasonable number of clusters, the number of mixture components of mixture Gaussian distribution is able to be estimated. Another advantage is that these parameters are estimated in on-line manner, because SOINN can accommodate on-line input patterns. These advantages of SOINN are available for modeling of time-series data. This paper introduces two SOINN based learning approaches for time-series data.

1.1 On-Line Supervised Learning Approach: SOINN-DTW

In [2][3], an supervised learning approach, SOINN-DTW, is proposed for recognition of time-series data, it is based on Dynamic Time Warping (DTW) and the Self-Organizing Incremental Neural Network (SOINN). In SOINN-DTW, probabilistic distance is used instead of Euclidean distance (local distance); a probabilistic path is used instead of a warping path. In addition, a time-series number of a template sequence corresponds to a number of a state, and a Gaussian distribution and kernel functions (Parzen window functions) are used for the output distribution. SOINN is used to estimate the output distribution. We do not need to define the number of mixture because SOINN is able to report the number of mixture-components. In addition, the output distribution is estimated in an on-line manner when time-series data is input incrementally. Therefore, SOINN-DTW is available as a technique for on-line learning tasks such as speaker adaptation.

1.2 On-Line Incremental Clustering Approach: HBSOINN

In [4], an Incremental unsupervised learning approach: HBSOINN is proposed for clustering of time-series data, the sequence patterns arising from human gestures. HMM contributes to robust feature extraction from time-series patterns, enabling similar statistical features to be extracted from time-series patterns of the same category. As a result of experiments with incremental clustering gesture data, we have found that HMM-based SOINN (HBSOINN) outperforms the other representative batch clustering approaches.

1.3 Related Works

A common approach for time-series data is the use of the Hidden Markov Model (HMM) [5], a generative model that includes hidden state structure. There has been much research on many researches of the theoretical analysis of HMM and a large amount of applications have been proposed based on HMM. We focus HMM-based approaches which are related to SOINN based approaches, and we briefly review the related research on HMM.

Supervised Learning Approaches. HMM is used for speech recognition [5] and motion recognition [6]. In HMM, time-series data is modeled using a Markov model that has finite states.

On-line learning techniques for HMM are proposed in [7][8][9]. However, these methods, which are based on HMM, require prior information that is attained through training of large amounts of batch data. Moreover, it is difficult for these methods based on HMM to learn (estimate) model parameters using on-line data without a large amount of batch data; it is also difficult for them to improve recognition performance continuously. In addition, when we use continuous-density HMM for on-line learning, we must choose the number of mixtures (M). If we choose a large value for M and only a small amount of on-line training

data, the model parameter over-fits the training data and recognition accuracy might therefore not be improved. If we choose a small value for M and a large amount of on-line training data is used, the generalization capability deteriorates and recognition accuracy might not be improved. Consequently, when we do not know the amount of on-line data, it is difficult to choose the optimal parameter M .

As [2] reported, using SOINN's unsupervised clustering function, the distribution of the output distribution is represented in a self-organizing manner; the number of states is set in the learning process. Therefore, we need not set prior conditions such as the number of states and the parameter of output distribution (e.g. the number of mixture M in mixture Gaussian distribution).

Unsupervised Learning Approaches. A well known clustering approach based on HMM is Mixture HMM clustering [10]. This approach assigns unlabeled training data to K clusters and K HMMs by EM framework. Another approach is spectral clustering [11]. This approach construct graphs based on the Kullback-Leibler distance between HMMs, and runs spectral clustering. These approaches are not able to cluster on-line input data and we need to set the number of clusters. On the other hand, HBSOINN enable on-line incremental clustering and reports the number of clusters incrementally.

[12] presents an incremental learning clustering approach for whole body motion patterns. Motion patterns are trained by HMM or factorial HMM [13] in an on-line manner. They are incrementally grouped by hierarchical agglomerative clustering and a hierarchy tree structure is generated automatically. Learned factorial HMM is used for both motion recognition and generation. This mechanism requires memorization of many HMMs that are abstracted from the input motion patterns. On the other hand, only typical prototype patterns are memorized in HBSOINN. Thus, HBSOINN can reduce the required system memory.

2 Algorithm1: SOINN-DTW

In SOINN-DTW, the global distance between training data is calculated using DTW. In addition, a template model is constructed based on DTW. Let N be the number of training data that belong to category \mathcal{C} . We can explain the construction procedure of the template model using N training data.

[Step 1: Selection of standard data]

Standard data P^* of the template model is selected from among N training data that belong to category \mathcal{C} . Standard data P^* is determined using the following equation.

$$P^* = \arg \min_{P_m} \left\{ \sum_{n=1}^N D(P_m, P_n) \right\} \quad (\{P_n, P_m\} \in \mathcal{C}) \quad (1)$$

In Equation (1), P_m, P_n denote training data which belong to category \mathcal{C} . In addition, $D(P_m, P_n)$ denotes the global distance in symmetric DTW [14], where T^* is the time length of standard data P^* .

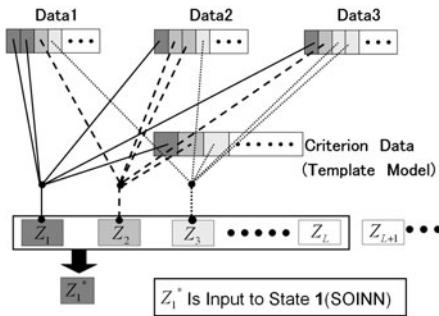


Fig. 1. Process of Step 2. (After DTW, the optimal path between the standard data and training data is determined. Corresponding data in the optimal path are input into each SOINN.)).

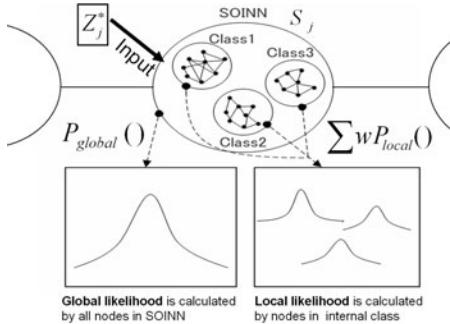


Fig. 2. Two probability distributions formed by the node set of SOINN

[Step 2: Allocation of samples to each state]

Let p_j^* be a sample of standard data \mathbf{P}^* at time j . Let p_i^n be a sample of training data \mathbf{P}_n ($n \in \mathcal{C}$) at time i . After DTW of the standard data \mathbf{P}^* and the training data, the optimal (warping) path between p_j^* and p_i^n is determined as the following equation, such that the global distance $D(\mathbf{P}^*, \mathbf{P}_n)$ is minimized.

$$i = w_j^n \quad (j = 1, 2, \dots, T^*) \quad (2)$$

Sample p_i^n at time i is divided into each state j of the template model according to $i = w_j^n$. This allocation of samples is done from time 1 to time T^* .

[Step 2] is executed for all training data ($n \in \mathcal{C}$). As a result, the $N - 1$ optimal path is determined as w^n ($n = 1, \dots, N - 1$). The allocation of samples is also done according to w^n . The set of samples allocated to each state j (SOINN) is defined as \mathcal{Z}_j .

Next, the set of samples \mathcal{Z}_j is input into SOINN. The topological structure of the distribution generated by \mathcal{Z}_j is represented after learning by SOINN. Samples in \mathcal{Z}_j are scarce when training data is scarce. The learning performance of SOINN worsens if samples that are included in \mathcal{Z}_j are scarce. A set of samples from \mathcal{Z}_j to \mathcal{Z}_{j+L-1} is input to the SOINN (state j) to prevent the problem. A set of samples that is allocated to each state j (SOINN) is defined as $\mathcal{Z}_j^* = \{\mathcal{Z}_j, \mathcal{Z}_{j+1}, \dots, \mathcal{Z}_{j+L-1}\}$ again, where L denotes the number of segments and is the parameter in SOINN-DTW. The state number is redefined as $T^* - L - 1$ in SOINN-DTW. Figure 2 portrays the process of Step 2.

[Step 3: Learning by SOINN]

After \mathcal{Z}_j^* is input into and learned by SOINN, the number of node sets (clusters) that are connected by edges is output by SOINN. The output distribution is estimated of SOINN (state j) from position vectors of these node sets A . The method of parameter estimation is described in [2].

The global distance between the template model TM_c of category c and input data IP is calculated using a recurrence formula. Local distance of DTW is defined as the probability value which is output from two kinds of probabilistic density functions (pdfs). The first pdf is defined as a multi variable gauss distribution and the second pdf as a Gaussian kernel function, and calculate the global/local likelihood by using the first/the second pdf (Fig. 2). In addition, SOINN-DTW enable on-line training by iterating Step 2 and Step 3.

2.1 Evaluation of SOINN-DTW

We explain the evaluation of batch and on-line learning performance of SOINN-DTW which are described in [2][3]. Parameters of SOINN-DTW are set as $\lambda = 10000$, $a_d = 10000$.

Evaluation of Batch Training. The phoneme classification experiments are performed, where the ked-TIMIT dataset and Resource management 1 (RM1) dataset are used as input. The ked-TIMIT dataset is composed of 39 phonemes and the dataset totals 3900 samples (100 data per 1 phoneme from single speaker). The RM1 dataset is composed of 27 phonemes and the dataset totals 3220 samples (120 data per 1 phoneme from 4 speakers).

After 10 experiments using two databases, the average classification rate is shown in Table II. Regarding abbreviations, [SO-DTW] denotes SOINN-DTW, [ST-DTW (1)] denotes Stochastic-DTW [15] that uses the asymmetric recurrence formula, and [ST-DTW (2)] denotes Stochastic-DTW that uses the symmetric recurrence formula. In Table II, $(\cdot S, \cdot M)$ in the classification rate of HMM denotes the parameter of the situation in which HMM has the best classification performance. In addition, S is the number of states and M is the number of mixtures.

Experiments were performed with changing the state's number $1 \leq N_{hmm} \leq 13$ and mixture's number M_{hmm} ($1 \leq M_{hmm} \leq 4$) to search optimal parameters (number of state, number of mixture), such that HMM had the best classification performance. Consequently, the classification rate is maximum, as presented in Table II.

According to Table II, the performance of the SOINN-DTW was clearly superior to that of either Stochastic-DTW or HMM, which had heretofore shown the best performance.

2.2 Evaluation of On-Line Training

An experiment using a phoneme data set is performed to evaluate the on-line learning performance. The MAP adaptation method is chosen as a comparative method.

About HMM based on the MAP adaptation method, the mean vector and mixture weights of HMM are adapted to on-line training data. When the number of on-line training data was 500, the recognition accuracy after on-line learning (adaptation) decreased more than that of the baseline. Furthermore, for HMM

Table 1. Classification rate in phoneme classification task [%] (TD40/TD80 denotes the number of training data)

	SO-DTW	ST-DTW (1)	ST-DTW (2)	HMM
k-TIMIT TD40[%]	56.36	30.81	51.71	47.69 (5S,2M)
k-TIMIT TD80[%]	62.55	33.83	55.46	51.90 (5S,4M)
RM1 TD80[%]	71.85	47.52	68.55	63.49 (3S,3M)

Table 2. The recognition accuracy of baseline and the recognition accuracy after on-line learning when the number of on-line training data are 3000 samples

	Baseline	On-line (+3000 data)
HMM-MAP (M1)	51.45	19.18
HMM-MAP (M5)	44.70	34.15
HMM-MAP (M10)	39.22	28.55
HMM-MAP (M15)	33.67	28.55
SOINN-DTW	41.25	56.73

based on MAP adaptation, the number of on-line training data was set to 1000, 3000, and 5000. Even when the number of on-line training data was also 1000, 3000, and 5000, the recognition accuracy after on-line learning decreased more than that of the baseline. Table 2 shows the baseline recognition accuracy and the recognition accuracy after on-line learning when the total number of on-line training data was 3000. With regard to HMM, the number of states was set to 3 and the number of mixtures was set to 1, 5, 10, and 15. For SOINN-DTW, each data was input to the model 500 samples at a time and learned incrementally, until the total number of on-line training data was 3000 (500×6 steps).

As shown in Table 2, in SOINN-DTW the recognition accuracy after on-line learning improved to about 15% more than that of the baseline. The result shows clearly that it is difficult for HMM based on MAP adaptation to improve the recognition accuracy when the amount of initial training data is few.

3 Algorithm2: HBSOINN

In order to input time-series data to SOINN, we need to map the variable length sequences to vectors of a fixed dimension. In HBSOINN, the HMM [5] is used to reduce the number of dimensions of the time-series data and to map the variable length sequences to vectors of a fixed dimension.

3.1 Hidden Markov Model for Mapping to Fixed Length Vector

The HMM that a left-to-right model was used based on the unit Gaussian probabilities with a diagonal-covariance matrix for each state. The model parameters are trained from training data using the Baum-Welch algorithm. In addition, to improve the parameter estimation performance, the segmental k-means algorithm is used for setting the initial parameters.

In this section, we explain how to extract features by HMM with N emitting states ($j \in \{1 \dots N\}$) and how to map time-series data to fixed length vectors. In this research, a motion capture system is used for gesture feature extraction. The time-series data is the gesture component of continuous motion data.

$$\mathcal{O}_k = \{o_{k,1}, o_{k,2}, \dots, o_{k,t}, \dots, o_{k,T_k}\} \quad (3)$$

k is the data number, $o_{k,t}$ is the dimensional feature vector, and T_k is the time series length. A training pattern O_k is trained by k th HMM. After training O_k , HMM parameters are used for input data to the SOINN. The HMM parameters are $\lambda_k = \{\pi_k, A_k, B_k\}$. In this equation, π_k is the initial probability, A_k is the transition probability, and B_k is the output probability. The output probability distribution B_k is formed by the Gaussian component $\mathcal{N}(\mu_{k,j}, \Sigma_{k,j})$. The component has a mean $\mu_{k,j}$ and covariance $\Sigma_{k,j}$. When all the parameters λ_k are input to SOINN, the input feature vector has many dimensions. This is known as the curse of dimensionality [16]. Thus, we need to choose sufficient features for time-series classification. In this research, the mean vector and covariance matrix of the output probability distribution are used for the input vector to SOINN because the output probability distribution of each state contains information of the trajectory of feature space. Here, only the mean vector is used for an input vector to avoid the curse of dimensionality.

Fig 3 shows the HMM mapping method. When the mean $\mu_{k,1}$ of state 1 is two dimensional vectors, the mean vector is $\mu_{k,1} = \{a_{k,1,1}, a_{k,1,2}\}$. When there are three states, the input vector ξ_k to SOINN, obtained from O_k , is 6 (2x3) dimensional vectors, is as follows:

$$\xi_k = \{\mu_{k,1}, \mu_{k,2}, \mu_{k,3},\} = \{a_{k,1,1}, a_{k,1,2}, a_{k,2,1}, a_{k,2,2}, a_{k,3,1}, a_{k,3,2}\} \quad (4)$$

Using HMM, the variable length sequence O_k is mapped to a vector of fixed dimension ξ_k .

3.2 Improvement of Incremental Learning Performance by Random Sampling

An algorithm of HBSOINN adds a random sampling function to that of SOINN. The random sampling is performed based on the output distribution of states.

If the distance between ξ_k and s_1 is greater than the similarity threshold Th_m , a random sampling approach is applied to generate sample vectors from k th HMM. The reason for performing this process is as follows. The input pattern ξ_k which satisfies the conditions is far from all existing nodes. The following processing is performed based on an assumption that the possibility that the input vector is an unknown pattern (gesture) is high. In order to learn the input vector frequency, k th HMM generates the vectors that are similar to the input pattern ξ_k .

N_R random samples $p_{k,j,l}(l \in \{1...N_R\})$ are generated from the multidimensional Gaussian distribution (output distribution) in each state j of k th HMM. When the number of states is N , and the number of dimensions D , the sample patterns $\xi'_{k,l}$ are defined as

$$\xi'_{k,l} = \{p_{k,1,l}, \dots, p_{k,N,l},\} \quad l \in \{1...N_R\} \quad (5)$$

As a result, the contents of the input pattern set are ξ_k and $\xi'_{k,l}(l \in \{1...N_R\})$. On the other hand, if the distance between ξ_k and s_1 is smaller than the similarity threshold Th_m , the input pattern is only ξ_k .

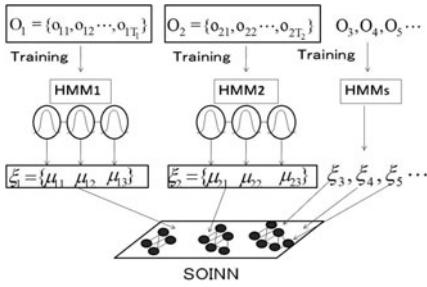


Fig. 3. Overview of HBSOINN (Method of converting a time-series pattern to a fixed length vector using HMM. This figure shows an HMM that has three states.)

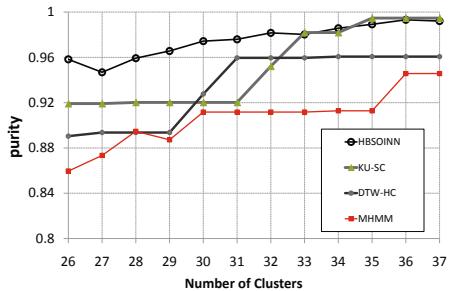


Fig. 4. Comparison of HBSOINN vs. batch clustering algorithms on the task 2

3.3 Evaluation of HBSOINN

The dataset is composed of 26 kinds of gestures and totals 940 samples. Parameters of HBSOINN are set as $\lambda = 100$, $a_d = 10$, $\sigma = 10000$, $Th_m = 30$, $N_R = 5$. A hierarchical clustering approach based on the DTW distance, spectral clustering [11] and Mixture HMM clustering [10] are used as the batch clustering approaches to compare with HBSOINN.

1000 experiments were performed, while changing the input pattern order. The distribution of the number of classes for HBSOINN was from 26 to 37. Figure 4 compares HBSOINN and batch clustering approaches in terms of the average purity [17] for between 26 and 37 output clusters. From Fig. 4, the purity of HBSOINN is superior to that of batch approaches except when the number of classes is 35 or 37. Thus, even though HBSOINN uses an incremental clustering approach, its clustering performance is superior or almost the same as the batch clustering approaches.

4 Future Works

Future work of SOINN-DTW is to implement the multi-template model. A classification parameter of multi-template SOINN-DTW will improve that of a single-template SOINN-DTW. Future work of HBSOINN is to estimate the number of HMM states and topology. Although the approach of setting the number or the topology of HMM has been proposed in previous works, it is difficult to set these parameters in an on-line manner. A future work of SOINN-DTW and HBSOINN is to extend to the approaches which enable them to apply to continuous motion or continuous speech. A preliminary approach toward the application to continuous time-series data is proposed in [18]. We plan to extend the approach [18] as the robust motion pattern discovery approach.

5 Conclusion

We introduced the machine learning algorithms: SOINN-DTW and HBSOINN of time-series data based on Self-organizing Incremental Neural Network (SOINN). From [2][3][4][18], it is shown that SOINN-DTW and HBSOINN are available for classification of time-series data such as phoneme and motion as well as clustering.

References

1. Shen, F., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. *Neural Networks* 19(1), 90–106 (2006)
2. Okada, S., Hasegawa, O.: Classification of temporal data based on selforganizing incremental neural network. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4669, pp. 465–475. Springer, Heidelberg (2007)
3. Okada, S., Hasegawa, O.: On-line learning of sequence data based on self-organizing incremental neural network. In: IEEE International Joint Conference on Neural Networks, IJCNN 2008 (IEEE World Congress on Computational Intelligence), pp. 3847–3854 (2008)
4. Okada, S., Nishida, T.: Incremental clustering of gesture patterns based on a self organizing incremental neural network. In: IEEE International Joint Conference on Neural Networks (IJCNN 2009) (2009)
5. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 257–286 (1989)
6. Wilson, A., Bobick, A.: Learning visual behavior for gesture analysis. In: Proc. IEEE International Symposium on Computer Vision, vol. 5A Motion2 (1995)
7. Gauvain, J.L., Lee, C.H.: Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing* 2(2), 291–298 (1994)
8. Leggetter, C., Woodland, P.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language* 9(2), 171–185 (1995)
9. Mongillo, G., Deneve, S.: Online learning with hidden markov models. *Neural computation* 20(7), 1706–1716 (2008)
10. Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V.: Discovering clusters in motion time-series data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 375–381 (2003)
11. Yin, J., Yang, Q.: Integrating hidden markov models and spectral analysis for sensory time series clustering. In: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), pp. 506–513 (2005)
12. Kulić, D., Takano, W., Nakamura, Y.: Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *Int. J. Rob. Res.* 27(7), 761–784 (2008)
13. Ghahramani, Z., Jordan, M.I., Smyth, P.: Factorial hidden markov models. In: Machine Learning. MIT Press, Cambridge (1997)

14. Rabiner, L., Juang, B.: *Fundamentals of Speech Recognition*. PTR Prentice-Hall, Inc., Englewood Cliffs (1993)
15. Nakagawa, S.: Speaker-independent consonant recognition in continuous speech by a stochastic dynamic time warping method. In: Proc. International Conference Pattern Recognition., vol. 70, pp. 925–928 (1986)
16. Duda, R., Hart, P., Stork, D.: *Pattern Classification*, 2nd edn. John Wiley Sons, Inc., Canada (2001)
17. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2007)
18. Okada, S., Ishibashi, S., Nishida, T.: On-line unsupervised segmentation for multidimensional time-series data and application to spatiotemporal gesture data. In: IEA/AIE 2010 (2010)

Online Knowledge Acquisition and General Problem Solving in a Real World by Humanoid Robots

Naoya Makibuchi¹, Furao Shen², and Osamu Hasegawa¹

¹ Department of Computational Intelligence and Systems Science,
Tokyo Institute of Technology, Yokohama 226-8503, Japan
{makibuchi.n.aa,hasegawa.o.aa}@m.titech.ac.jp
<http://www.haselab.info/>

² The State Key Laboratory for Novel Software Technology,
Nanjing University, China
frshen@nju.edu.cn

Abstract. In this paper, the authors propose a three-layer architecture using an existing planner, which is designed to build a general problem-solving system in a real world. A robot, which has implemented the proposed method, forms the concepts of objects using the Self-Organizing Incremental Neural Network, and then acquires knowledge, online and incrementally, through interaction with the environment or with humans. In addition, it can solve general-purpose problems in a real world by actively working with the various acquired knowledge using the General Problem Solver. In the experiment, the authors show that the proposed method is effective for solving general-purpose problems in a real world using a humanoid robot.

Keywords: Real World Intelligence, Humanoid Robot, Self-Organizing Incremental Neural Network, General Problem Solver.

1 Introduction

To enable humanoid robots to act intelligently in a real world, not only a control theory for robot motion or an information theory for robot intelligence, but also a close collaboration of various fields of study is needed. For the field of motion control of humanoid robots, many studies have been vigorously performed in private enterprises or research organizations, such as the study of an autonomy bipedal robot represented by ASIMO of Honda Motors, or the study of object-handling learning by a small humanoid robot [1]. However, on the other hand, in the field of intelligent information processing of humanoid robots, studies that can say “able to execute non-programmed operations autonomously” have not been reported so far. Therefore, this study proposes an architecture based on a keyword, that is, the intelligence of humanoid robots, specifically used for general-purpose problem solving in a real world.

Designers have traditionally created intelligent robots by incorporating any possible situations and actions in advance. However, for a real world that continuously changes in a complex way, it is impossible to incorporate appropriate actions every time in advance. Instead of such a conventional method, there is an approach called Cognitive Developmental Robotics (CDR) [2] that focuses on robots' embodiment, in other words, it uses a process where the robots make contact with the environment through their own bodies, and then understand the information obtained from a real world. On the other hand, in terms of robot intelligence development, Weng and his followers have concluded that the following properties are required for a developmental system [3]:

- I. The system is not specific to tasks.
- II. The tasks are unknown to the system designers.
- III. The system can generate approaches for unknown tasks.
- IV. The system has an online learning ability.
- V. The system has an open-ended learning ability.

An approach based on CDR could meet all the properties, except for property III because this property is obviously different from the others in terms of the intelligence level required. In particular, it demands not only the ability to "acquire" knowledge (solutions for tasks that are then expressed as knowledge), but also the ability to "generate" knowledge autonomously. To achieve this, in addition to the basic concept of CDR, a mechanism to generate new knowledge from existing knowledge is needed.

1.1 Purpose of This Study

The purpose of this study is to build a general problem-solving system in a real world that meets all the properties, including property III as defined by Weng. The proposed method is an architecture shown in Figure 1; a robot implemented using this architecture displays the following features:

- The robot forms the concepts of objects (symbols) from patterns obtained by seeing and hearing.
- The robot can acquire cause-effect relationships between actions and environmental changes in form of knowledge, both online and incrementally, through interaction with the environment or with humans.
- The robot can generate approaches for unknown tasks by combining existing knowledge.

2 Proposed Method

The architecture of the proposed method is shown in Figure 1. It is a three-layer architecture consisting of an Input Layer, a Pattern Memory Layer, and a Symbol Memory Layer. Among these, the Input Layer consists of three phases: a Symbol Grounding Phase, a Knowledge Acquisition Phase, and a Problem Solving Phase.

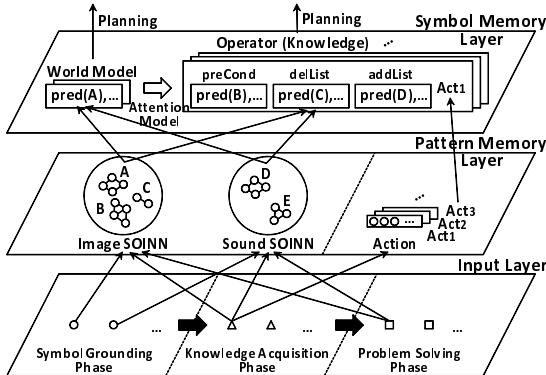


Fig. 1. Architecture of the proposed method. The black arrow in the Input Layer indicates the flow of processing. Arrows between each layer indicate the flow of data. A white arrow in the Symbol Memory Layer signifies the generation of operators (knowledge). In addition, a border in the Pattern Memory Layer means that the robot's movement information is kept in a space separate from the SOINN spaces.

In these phases, image and sound patterns are input from real environments. In addition, concept information added to each pattern in the Symbol Grounding Phase and the time-series data of the robot's joint angles (Act1, Act2, Act3 ...) in the Knowledge Acquisition Phase are input. In Figure 1, a circle, a triangle and a square in the Input Layer means that the type of each pattern is different by phases, as described. The Pattern Memory Layer stores patterns sent from the Input Layer, so that it functions as an interface for mapping a pattern to a symbol. The Symbol Memory Layer retains World Models and Operators, which include predicates (pred) that get symbols (A, B, ..) as arguments from the Pattern Memory Layer, which are used for planning.

1. **Symbol Grounding Phase** — In this phase, a robot forms the concepts of image and sound objects, which are used in later phases, using input patterns from real environments. This is realized by employing the Self-Organizing Incremental Neural Network (SOINN) [5]. The SOINN is capable of online incremental learning, that is, it can learn new input data without both storing all the previous ones and destroying the old learned knowledge. In this study, we prepare two SOINN spaces (Image SOINN and Sound SOINN) for image and sound patterns in the Pattern Memory Layer. A robot inputs each pattern to each SOINN along with the concept information provided by the experimenters.
2. **Knowledge Acquisition Phase** — In this phase, a robot acquires, both online and incrementally, cause-effect relationships in the form of knowledge from the changes to real environments that have been caused by its own actions. These cause-effect relationships are expressed, respectively, as one operator by the Attention Model. In particular, the Attention Model constructs an

operator by obtaining three elements necessary for its composition (a precondition (preCond), a deletion list (delList), and an addition list (addList)) from the changes to a real environment and then by combining these elements with taught movement information (Act).

3. Problem Solving Phase — In this phase, a robot solves general-purpose problems in real environments by working operators acquired in the Knowledge Acquisition Phase. This is achieved by employing the General Problem Solver (GPS) [4], which is a well-known planner for actively approaching working knowledge. First, a robot judges whether there are appropriate approaches for a task presented in a real environment. If such approaches exist, the robot chooses an approach according to a constant evaluation criterion, and then executes the approach in real environments. In this case, the robot repeats executing an operator and observing the changes to a real environment until it reaches a goal state.

3 Experiments

In this experiment, a robot acquires operators, both online and incrementally, as shown in Figure 2, such as, “If the robot raises its right/left hand, an apple/a bell will be put in his hand,” “If the robot presses the bell, it will ring,” and “If the robot makes a ‘give me’ gesture while the bell is ringing, the apple will move in front of it.” We then presented tasks to the robot such as, “Put an apple in front of it when there is nothing on the table,” which it had not directly experienced previously. As a consequence, as shown in Figure 3, the robot could perform the tasks by combining existing knowledge without direct teaching from the experimenters.



Fig. 2. An example of operators that the robot acquired

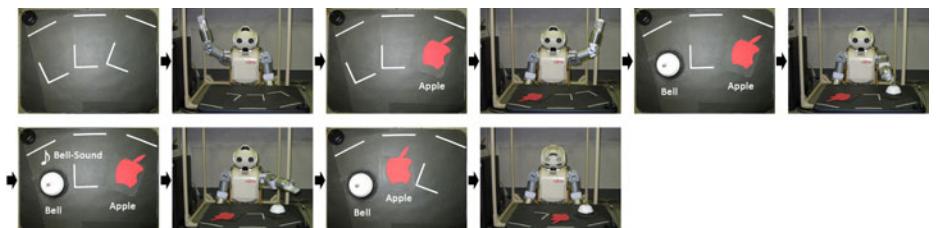


Fig. 3. An example of tasks that the robot performed. The last figure shows the robot nodding its head with satisfaction that it was able to reach a goal state.

4 Discussions

The proposed method meets all the properties, including property III as proposed by Weng, so that a robot implemented with it can act appropriately for tasks that it has not experienced directly, using actively acquired knowledge.

The SOINN is a key technology of the proposed method. Regarding only the symbolization of sense data, approaches using the Self-Organizing Map (SOM), the Recurrent Neural Network with Parametric Bias (RNNPB), or the Hidden Markov Model (HMM) are also conceivable (e.g., [6,7,8]). However, because it is generally necessary to define their internal structure beforehand, such as the number of nodes or states, they are not appropriate for online incremental learning. Although some of these approaches are, in principle, capable of online incremental learning, in fact, the performance of incremental learning using the SOM or RNNPB depends considerably on the number of nodes composing the network. Besides, since statistical methods such the HMM need a large amount of learning data, these methods are undesirable for robots in a real world from a practical standpoint. On the other hand, in an online incremental learning using the SOINN, not only it is unnecessary to define its size in advance, but also adaptation capability in a self-organizational way and high robustness to noise can be expected. These properties are essential to meet all the properties proposed by Weng.

Because it employs a layer structure that includes SOINNs, following are some superior aspects of the proposed method:

- It is capable of behaving robustly in the handling of unstable patterns in a real world. In addition, the concepts of objects can be also formed online and incrementally, which is necessary for knowledge acquisition and problem solving.
- It is capable of multimodal information processing required by robots, using multiple senses such as sight and sound. This study deals only with image and sound data, but in fact, other sensory data can be easily incorporated by preparing another SOINN space in the Pattern Memory Layer.
- A pattern set expressing the concepts of objects is not kept in individual operators but in the Pattern Memory Layer. Hence, concepts formed in the Pattern Memory Layer can be shared as a symbol from the Symbol Memory Layer, which results in significant savings in the memory capacity of operators in the Symbol Memory Layer.

5 Conclusion and Future Work

In this paper, the authors propose a three-layer architecture using an existing planner, which is designed to build a general problem-solving system in a real world. We consider this study as an important basis for the creation of humanoid robots that act intelligently in a real world. In terms of future studies, we will mainly focus on discussions of the following topics:

1. Automatic actions by getting used to tasks — In the current method, the more the number of operators acquired, the longer the time taken to plan. Namely, a robot that has a large amount of knowledge does not act at once when a task is presented to it. Therefore, by extending the architecture, we are now discussing a mechanism to cope automatically with the tasks that a robot has experienced before, without involving a planner.
2. Task instructions using languages — In the current method, task instructions are performed by presenting the goal states of tasks to a robot in real environments. However, it is paradoxical that we must prepare the goal states in advance of tasks that we want the robot to perform. Therefore, we are now discussing a mechanism to give the goal states of tasks to a robot using languages.

Acknowledgment

This study was supported by the Industrial Technology Research Grant Program in 2010 from the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

1. Ito, M., Noda, K., Hoshino, Y., Tani, J.: Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks* 19, 323–337 (2006)
2. Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., Yoshida, C.: Cognitive Developmental Robotics: A Survey. *IEEE Trans. Autonomous Mental Development* 1(1), 12–34 (2009)
3. Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., Thelen, E.: Autonomous mental development by robots and animals. *Science* 291(5504), 599–600 (2001)
4. Ernst, G.W., Newell, A.: *GPS: A Case Study in Generality and Problem Solving*. Academic Press, New York (1969)
5. Shen, F., Hasegawa, O.: An Incremental Network for On-line Unsupervised Classification and Topology Learning. *Neural Networks* 19, 90–106 (2006)
6. Minamino, K., Aoyama, K., Shimomura, H.: Voice Imitation based on self-organizing maps with HMMs. In: Proc. International Workshop on Intelligence Dynamics at Humanoids, pp. 24–29 (2005)
7. Inamura, T., Shibata, T.: Interpolation and Extrapolation of Motion Patterns in the Proto-symbol Space. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) *ICONIP 2007, Part II. LNCS*, vol. 4985, pp. 193–202. Springer, Heidelberg (2008)
8. Ogata, T., Matsumoto, S., Tani, J., Komatani, K., Okuno, H.G.: Human-Robot Cooperation using Quasi-symbols Generated by RNNPB Model. In: Proc. IEEE International Conference on Robotics and Automation, pp. 2156–2161 (2007)

Incremental Learning Using Self-Organizing Neural Grove

Hirotaka Inoue

Department of Electrical Engineering and Information Science,
Kure National College of Technology,
2-2-11 Agaminami, Kure-shi, Hiroshima, 737-8506 Japan
hiro@kure-nct.ac.jp

Abstract. Recently, multiple classifier systems (MCS) have been used for practical applications to improve classification accuracy. Self-generating neural networks (SGNN) are one of the suitable base-classifiers for MCS because of their simple setting and fast learning. However, the computation cost of the MCS increases in proportion to the number of SGNN. We proposed a novel pruning method for efficient classification and we call this model as self-organizing neural grove (SONG). In this paper, we investigate SONG's incremental learning performance.

1 Introduction

Classifiers need to find hidden information in the given large data effectively and classify unknown data as accurately as possible [1]. Recently, to improve the classification accuracy, multiple classifier systems (MCS) such as neural network ensembles, bagging, and boosting have been used for practical data mining applications [2]. In general, the base classifiers of the MCS use traditional models such as neural networks (backpropagation network and radial basis function network) [3] and decision trees (CART and C4.5) [4].

Neural networks have great advantages of adaptability, flexibility, and universal nonlinear input-output mapping capability. However, to apply these neural networks, it is necessary to determine the network structure and some parameters by human experts, and it is quite difficult to choose the right network structure suitable for a particular application at hand. Moreover, they require a long training time to learn the input-output relation of the given data. These drawbacks prevent neural networks being the base classifier of the MCS for practical applications.

Self-generating neural networks (SGNN) [5] have simple network design and high speed learning. SGNN are an extension of the self-organizing maps (SOM) of Kohonen [6] and utilize the competitive learning which is implemented as a self-generating neural tree (SGNT). The abilities of SGNN make it suitable for the base classifier of the MCS.

In an earlier paper [7], we proposed a pruning method for the structure of the SGNN in the MCS to improve the generalization capability and reduce the

computation cost. We call this model as self-organizing neural grove (SONG). This pruning method is constructed from two stages. At the first stage, we introduce an on-line pruning algorithm to reduce the computation cost by using class labels in learning. At the second stage, we optimize the structure of the SGNT in the MCS to improve the generalization capability by pruning the redundant leaves after learning. In the optimization stage, we introduce a threshold value as a pruning parameter to decide which subtree's leaves to prune and estimate with 10-fold cross-validation [8]. After the optimization, the SONG can improve its classification accuracy as well as reducing the computation cost. We use bagging [9] as a resampling technique for the SONG.

In this paper, we investigate the incremental learning performance of the SONG using a problem in the UCI machine learning repository [10]. The rest of the paper is organized as follows: the next section shows how to construct the SONG. Then section 3 is devoted to some experiments to investigate the incremental learning performance of SONG. Finally we present some conclusions, and outline plans for future work.

2 Constructing Self-Organizing Neural Grove

In this section, we describe how to prune redundant leaves in the SONG. First, we mention the on-line pruning method in learning of SGNT. Second, we show the optimization method in constructing the SONG.

2.1 On-Line Pruning of Self-Generating Neural Tree

SGNN are based on SOM and implemented as an SGNT architecture. The SGNT can be constructed directly from the given training data without any intervening human effort. The SGNT algorithm is defined as a tree construction problem of how to construct a tree structure from the given data which consist of multiple attributes under the condition that the final leaves correspond to the given data.

Before we describe the SGNT algorithm, we denote some notations.

- input data vector: $e_i \in \mathbb{R}^m$.
- root, leaf, and node in the SGNT: n_j .
- weight vector of n_j : $w_j \in \mathbb{R}^m$.
- the number of the leaves in n_j : c_j .
- distance measure: $d(e_i, w_j)$.
- winner leaf for e_i in the SGNT: n_{win} .

The SGNT algorithm is a hierarchical clustering algorithm. The pseudo C code of the SGNT algorithm is given in Fig. 1. In Fig. 1, several sub procedures are used. Table 1 shows the sub procedures of the SGNT algorithm and their specifications.

Input:

```
A set of training examples E = {e_i},
i = 1, ..., N.
A distance measure d(e_i, w_j).
```

Program Code:

```
copy(n_1, e_1);
for (i = 2, j = 2; i <= N; i++) {
    n_win = choose(e_i, n_1);
    if (leaf(n_win)) {
        copy(n_j, w_win);
        connect(n_j, n_win);
        j++;
    }
    copy(n_j, e_i);
    connect(n_j, n_win);
    j++;
    prune(n_win);
}
```

Output:

Constructed SGNT by E.

Fig. 1. SGNT algorithm

Table 1. Sub procedures of the SGNT algorithm

Sub procedure	Specification
<i>copy(n_j, e_i/w_{win})</i>	Create n _j , copy e _i /w _{win} as w _j in n _j .
<i>choose(e_i, n₁)</i>	Decide n _{win} for e _i .
<i>leaf(n_{win})</i>	Check n _{win} whether n _{win} is a leaf or not.
<i>connect(n_j, n_{win})</i>	Connect n _j as a child leaf of n _{win} .
<i>prune(n_{win})</i>	Prune leaves if the leaves have the same class.

In order to decide the winner leaf n_{win} in the sub procedure `choose(e_i, n_1)`, the competitive learning is used. This sub procedure is recursively used from the root to the leaves of the SGNT. If an n_j includes the n_{win} as its descendant in the SGNT, the weight w_{jk} ($k = 1, 2, \dots, m$) of the n_j is updated as follows:

$$w_{jk} \leftarrow w_{jk} + \frac{1}{c_j} \cdot (e_{ik} - w_{jk}), \quad 1 \leq k \leq m. \quad (1)$$

After all training data are inserted into the SGNT as the leaves, the leaves have each class label as the outputs and the weights of each node are the averages of the corresponding weights of all its leaves. The whole network of the SGNT reflects the given feature space by its topology. For more details concerning how to construct and perform the SGNT, see [5]. Note, to optimize the structure of the SGNT effectively, we remove the threshold value of the original SGNT algorithm in [5] to control the number of leaves based on the distance because

```

1 begin   initialize  $j =$  the height of the SGNT
2   do for each subtree's leaves in the height  $j$ 
3     if the ratio of the most class  $\geq \alpha$ ,
4       then merge all leaves to parent node
5     if all subtrees are traversed in the height  $j$ ,
6       then  $j \leftarrow j - 1$ 
7   until  $j = 0$ 
8 end.

```

Fig. 2. The merge phase

```

1 begin initialize  $\alpha = 0.5$ 
2   do for each  $\alpha$ 
3     evaluate the merge phase with 10-fold CV
4     if the best classification accuracy is obtained,
5       then record the  $\alpha$  as the optimal value
6        $\alpha \leftarrow \alpha + 0.05$ 
7   until  $\alpha = 1$ 
8 end.

```

Fig. 3. The evaluation phase

of the trade-off between the memory capacity and the classification accuracy. In order to avoid the above problem, we introduce a new pruning method in the sub procedure `prune(n_win)`. We use the class label to prune leaves. For leaves that have the n_{win} 's parent node, if all leaves belong to the same class, then these leaves are pruned and the parent node is given the class.

2.2 Optimization of the SONG

The SGNT has the capability of high speed processing. However, the accuracy of the SGNT is inferior to the conventional approaches, such as nearest neighbor, because the SGNT has no guarantee to reach the nearest leaf for unknown data. Hence, we construct the SONG by taking the majority of plural SGNT's outputs to improve the accuracy.

Although the accuracy of the SONG is superior or comparable to the accuracy of conventional approaches, the computational cost increases in proportion to the increase in the number of SGNTs in the SONG. In particular, the huge memory requirement prevents the use of SONG for large datasets even with latest computers.

In order to improve the classification accuracy, we propose an optimization method of the SONG for classification. This method has two parts, the merge phase and the evaluation phase. The merge phase is performed as a pruning algorithm to reduce dense leaves (Fig. 2).

This phase uses the class information and a threshold value α to decide which subtree's leaves to prune or not. For leaves that have the same parent node, if the

proportion of the most common class is greater than or equal to the threshold value α , then these leaves are pruned and the parent node is given the most common class.

The optimum threshold values α of the given problems are different from each other. The evaluation phase is performed to choose the best threshold value by introducing 10-fold cross validation (Fig. 3).

3 Experimental Results

In this section, we investigate the performance of the incremental learning of the SONG. We use letter in this experiment since it contains large scale data (the number of input dimension: 16, the number of classes: 26, and the number of entries: 20000).

This experiment is performed as follows. First, we divide letter dataset into ten parts. Second, we select one of the ten parts as the testing data. Third, we enter one of the remaining nine parts to the SONG for training. Forth, we test the SONG using the testing data. Finally, we continue the training and the testing until all nine parts dataset is entered the SONG.

Fig. 4 (a) shows the relation between the number of training data and the classification accuracy. The more the number of training data increases, the more the classification accuracy improves for all the number of ensembles K . The width of the improvement is wide in small K for all the number of N .

As the memory requirement, we count the number of units which is the sum of the root, nodes, and leaves of the SGNT. Fig. 4(b) shows the relation between the number of training data N and the number of units in $\alpha = 1$. Here, the total units are the number of all units without pruning and the remaining units are the number of all units with pruning. Both of them are the average of 25 SGNTs. The number of nodes increases linearly in proportion to increase the number of training data. The slope of the remaining units are smaller than the slope of the total nodes. It means that the SONG has a capability of good compression for

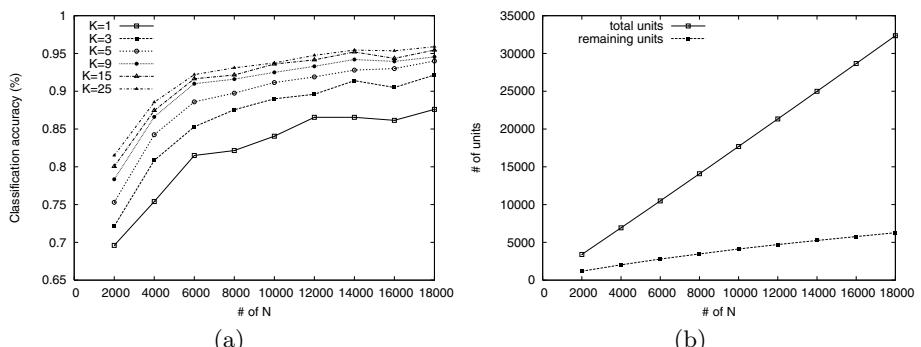


Fig. 4. The relation between the number of training data and (a) the classification accuracy, and (b) the number of units

large scale data. This supports that the SONG can be effectively used for large scale datasets.

4 Conclusions

In this paper, we investigated an incremental learning performance of SONG. We introduced an on-line and off-line pruning method and evaluated the SONG by 10-fold cross-validation. Experimental results showed that the memory requirement reduces remarkably, and the accuracy increases by using the pruned SGNT as the base classifier of the SONG. The SONG is a useful and practical MCS to classify large datasets. In future work, we will study a parallel and distributed processing of the SONG for large scale data mining.

References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2000)
2. Quinlan, J.R.: Bagging, Boosting, and C4.5. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR, August 4-8, pp. 725–730. AAAI Press, The MIT Press (1996)
3. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, New York (1995)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons Inc., New York (2000)
5. Wen, W.X., Jennings, A., Liu, H.: Learning a neural tree. In: The International Joint Conference on Neural Networks, Beijing, China, November 3–6, vol. 2, pp. 751–756 (1992)
6. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1995)
7. Inoue, H.: Self-Organizing Neural Grove: Efficient Multiple Classifier System with Pruned Self-Generating Neural Trees. In: Franco, L., Elizondo, D.A., Jerez, J.M. (eds.) Constructive Neural Networks. SCI, vol. 258, pp. 281–291. Springer, Berlin (2009)
8. Stone, M.: Cross-validation: A review. Math. Operationsforsch. Statist., Ser. Statistics 9(1), 127–139 (1978)
9. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
10. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2007),
<http://www.ics.uci.edu/mlearn/MLRepository.html>

Fast and Incremental Attribute Transferring and Classifying System for Detecting Unseen Object Classes

Aram Kawewong and Osamu Hasegawa

4259-R2-527 Nagatsuta, Midori-ku, Yokohama, Japan

Imaging Science and Engineering Laboratory

Tokyo Institute of Technology

{kawewong.a.aa, hasegawa.o.aa}@m.titech.ac.jp

Abstract. The problem of object classification when training and test classes are completely disjoint has recently become very popular in computer vision. To solve such problem, one needs to find common attributes of object and transfer them to use in classifying unseen object classes. Unfortunately, most recent attribute classifiers require the full batch-learning. This harshly prohibits an unseen-objects detection system from being used in online incremental machine such as robotics. This paper introduces a new approach for learning and classifying object's attribute in an online incremental manner. An approach is based on Self-Organizing and Incremental Neural Networks (SOINN). The evaluation has been done with 50-classes animal image dataset (30,000+ images). Comparing to the state-of-the-art method, our proposed approach named AT-SOINN (Attribute Transferring based on SOINN) performs the fast attribute learning, transferring and classification, while retaining high accuracy of attribute classification. Proposing AT-SOINN advances one more step towards online incremental unseen-object detections.

Keywords: Attribute Transferring, Self-Organizing and Incremental Neural Networks, Object Classification.

1 Introduction

Object recognition and classification in natural images have made large progress over the last years. For the task of specific object detection such as faces or vehicles, very powerful detectors and recognizers have been available by the combination of very distinctive low-level features, e.g. SIFT or SURF, and modern machine learning mechanism like support vectors machine (SVM). Nevertheless, to obtain good accuracy, a large number of manually labeled training data are needed on a regular basis, typically hundreds or thousands of sample images for each individual class to be learned. This strongly motivates current researchers to think about detectors that are capable of detecting a new unseen object class.

Up to date, transferring the attribute between classes lay emphasis on better generalization performance, typically for object classes with only few available training samples. At the very beginning, although the methods allow attribute or knowledge transferring between classes, all of them require at least some training samples and

cannot handle completely new object classes [1], [2], [3]. The exceptions are [5], [6], [7] where high-level attributes are used to learn descriptors of objects. These methods are capable of detecting the completely unseen object classes. Most recently, Farhardi et al. [8] set the problem to be more interesting and specific in which objects are annotated in detail. Although this method requires more effort of human supervision, the outcome is consequently significant in which the system not only recognizes the new unseen object classes but also their positions, view-points and some corresponding multi-level attributes.

Despite great success, all of the methods described previously aim at only the improvement of generalization; they mostly ignore the computation time and the flexibility of use in various kind of application. Considering the work of [7] as an example, the method needs to train the attribute classifier for each individual attribute. In testing stage, each attribute classifier will predict the probability for each attribute and the final probability score is calculated based on Bayes' theory. Each attribute classifier is trained by SVM which requires several hours for training. Doing this for all attributes (i.e. 85 attributes) would be too exhausted to use in robotics and other online application. Also, new input training data cannot be learnt incrementally because re-training all SVMs for all attributes is impractical.

Our aim is to create the attribute classifier that run in an incremental online manner while retaining good accuracy for recognition. Based on the Self-Organizing and Incremental Neural Networks (SOINN) [4], we create a novel Attribute Transferring system based on SOINN, coined as AT-SOINN. For each attribute, we create two individual SOINNs to be the positive-sample and negative-sample classifiers. In testing, the system performs the classification by considering the ratio of distance between nearest-positive-cluster and nearest-negative-cluster. The performance of AT-SOINN is evaluated by the same standard dataset of [7] under the same condition. While each attribute classifier of [7] requires several hours for training, our proposed method requires only 10-15 minutes. In addition, our method can learn the new data incrementally, while the resulting accuracy is still sufficiently high.

2 Briefly on SOINN

Self-Organizing and Incremental Neural Networks proposed by [4] is a mechanism for online unsupervised classification learning. Starting from the empty set, SOINN select the first two input data as the starting two nodes. Then, if the distances between the new input pattern and the first and second winner are less than the threshold, the pattern will be assigned to the first winner node. Otherwise, the SOINN will determine that the input pattern is too different from the current nodes and so that a new node should be created.

Consider the case in which a new pattern is assigned to nearest node s_1 in the SOINN, the weight vector W_{s_1} will be updated by the value of the new input pattern. Also, the edge between the first and the second winner will be created (if not existed). This renders SOINN significantly different from other clustering method such as K-Mean. The new input pattern or data is not directly added to the form the cluster. Instead, the cluster is formed by connecting the existing node in the SOINN. This

greatly saves much memory for running in long-term; a new node would be created if and only if the input pattern is significantly different from the current nodes in SOINN. Another key feature of SOINN is giving node some important properties. This idea is to make a node like an autonomous agent. A node has age, accumulated error, etc. As a result, at any time, each node has its own age, the accumulated error of the node (represented by accumulating the distance of input pattern for every time it is the first winner). With these properties, each node can perform two activities: being dead and dividing itself. If the node exist for a long time without winning for any new input patterns (noisy or useless node), all connected edges will gradually die. In addition, if the accumulated noise is too great, the node will divide itself into two. This resembles the action of recursively K-Mean for some big cluster.

3 AT-SOINN

In this section, we describe in detail about the proposed AT-SOINN. For ease of understanding, we should first describe again about the nature of the problem we want to solve by AT-SOINN.

3.1 Problem Definition

Actually, the problem of attribute classification is a sub-problem of unseen object detection. For the reader to clearly understand the problem and also the motivation of why we want to solve the attribute classification problem, we should first describe about the whole unseen object detection problem and then the sub-problem of attribute classification. Based on the Direct-Attribute-Prediction model used in [7], let $(x_1, l_1), \dots, (x_n, l_n) \subset \mathcal{X} \times \mathcal{Y}$ be training samples where \mathcal{X} is an arbitrary feature space and $\mathcal{Y} = \{y_1, \dots, y_K\}$ consists of K discrete classes, while $\mathcal{Z} = \{z_1, \dots, z_L\}$ is a test set of classes being disjoint from \mathcal{Y} . The main task is to learn a classifier $f: \mathcal{X} \rightarrow \mathcal{Z}$ for a label set $\mathcal{Z} = \{z_1, \dots, z_L\}$ that is completely disjoint from \mathcal{Y} .

Given the situation of learning with disjoint training and test classes, the attribute-based classification problem is defined. If for each class $z \in \mathcal{Z}$ and $y \in \mathcal{Y}$ an attribute representation $a \in \mathcal{A}$ is available, then a non-trivial classifier $\alpha: \mathcal{X} \rightarrow \mathcal{Z}$ can be learnt by transferring information between \mathcal{Y} and \mathcal{Z} through \mathcal{A} .

For simplicity, it is assumed that all attributes have binary values such that the attribute representation $a^y = (a_1^y, \dots, a_m^y)$ for any training class y is fixed-length binary vectors. The learning process starts by learning probabilistic classifiers for each attribute a_m . All images from training classes set \mathcal{Y} are used as training samples with their label determined by the entry of the attribute vector corresponding to the sample's label, i.e. a sample of class y is assigned the binary label a_m^y . The trained attribute classifiers provide us with estimates of $p(a_m|x)$, from which a model for the complete image-attribute layer as $p(a|x) = \prod_{m=1}^M p(a_m|x)$. This estimated term is used in calculating the posterior of a test class given an image:

$$p(z|x) = \sum_{a \in \{0,1\}^M} p(z|a)p(a|x) = \frac{p(z)}{p(a^z)} \prod_{m=1}^M p(a_m^z|x) \quad (1)$$

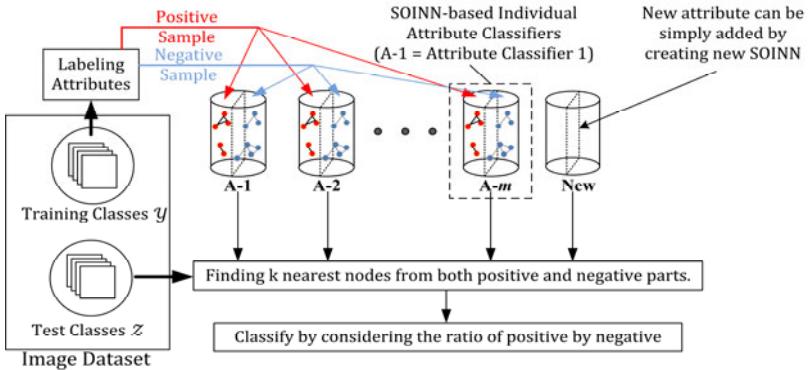


Fig. 1. Illustration of the AT-SOINN framework

Patches	Swims	Walks	Forest	Furry	Paws

Fig. 2. Highest ranking results for typically selected attributes. A red box indicates the mistakes. The classifier of attribute ‘patch’ obtain AUC = 0.65 while [7] obtained only 0.51.

By the equation, it is clear that the recognition performance directly depends on the quality of classifier of each attribute to estimate $p(x|a)$. In our study, we focus on creating the *online incremental attribute classifier* because the probability score obtained from each classifier can be finally combined based on Bayes’ rule to do recognition. Thus, our task here is to create the classifier of each attribute which is (1) online incremental, (ii) sufficiently fast for online application and (iii) accurate.

3.2 Proposed Method

The SOINN itself performs like the online incremental clustering tool. Generally, SOINN has been used in multi-class classification [4]. However, the main task here is to answer if an image contains an individual attribute. Thus, the problem is the binary classification. Additionally, the number of classes is fixed; there are only positive (+) class and negative (-) class. Thus we divide the SOINN into two parts: positive and negative parts. In each part, we let the cluster grows incrementally like the original

SOINN [4]. Since one SOINN is needed for one classifiers, totally m SOINNs are needed for classifying m attributes of images.

Figure 1 illustrates the framework of AT-SOINN. In training stage, training images from training classes are gradually input to the m SOINNs in an incremental manner. The attribute labeling module is presented here in case the system is to be used with robots. The supervisor (human) labels attribute of each image class via this module. In our study, the attribute labels are obtained from [7]. By the labels, positive and negative samples for training each attribute classifier can be separated. The learning can be run incrementally. When we want to test the AT-SOINN system, we input the test image of unseen image class to the system. Being represented by a feature vector, a test image x is input to every individual SOINN. Note that our used dataset here is the pre-extracted image data (please refer [7] for the detail of image feature extraction and test/train data separation). For an individual attribute classifier i , the sets of k nearest nodes from both positive part $\mathcal{S}_i^+ = \{s_{i,1}^+, \dots, s_{i,k}^+\}$ and negative part $\mathcal{S}_i^- = \{s_{i,1}^-, \dots, s_{i,k}^-\}$ are obtained. Then, the classification of the input image x as containing attribute i if and only if the following statement hold true.

$$\frac{\sum_{j=1}^k \|\xi - W_{i,j}^+\|}{\sum_{j=1}^k \|\xi - W_{i,j}^-\|} < T \quad (2)$$

where ξ is the input pattern and $W_{i,j}^+$ is the weight vector of the node $s_{i,j}^+$.

3.3 Evaluation on 30,000+ Animal Image Dataset

We obtain the Animals with Attributes dataset from [7]. The dataset contains 85 attributes with 50 animal classes. Instead of raw image data, we obtain the pre-extracted feature dataset so that we can make a fair comparison between their method and ours. Out of six different image features selected and used in [7], we use *only* SIFT feature for our attribute classification task. Although AT-SOINN can learn incrementally, we select to test its performance when 40 image classes have been learned. That is, 10 disjoint classes are used for testing the performance of AT-SOINN. This condition is exactly the same as done by [7] so this allows the fair comparison between AT-SOINN and [7]. For SOINN's parameter, we set the parameter as follows: $\text{age}_{\text{dead}} = 100$, $\lambda = 250$. Other parameters have been used as same as those set by [4].

Figure 3(c) shows the quality of the individual attribute classifiers of AT-SOINN comparing to [7]. Even though the accuracy of many attributes is lower than the method of [7], they are sufficiently high considering the drastically reduced computation time shown in the Fig. 3(a). In aspect of time, each classifier requires approximately 300 seconds for training on 40 image classes (24294 images) while the method of [7] requires *several hours* (thus totally more than ~ 100 hrs. are needed for training all attribute classifiers). Also, unlike other methods, new input images can be input *incrementally* without needs to restart the whole learning mechanism because AT-SOINN is fully incremental. Figure 2 shows result of some sample attributes. While these classifiers are trained on disjoint different class of animals, they can efficiently describe about the attribute of new previously unseen animal classes (i.e. the system can answer that giant-panda have a paw without needs to see any pictures of giant-panda). Sample of ROC curves are shown in Fig. 3(c).

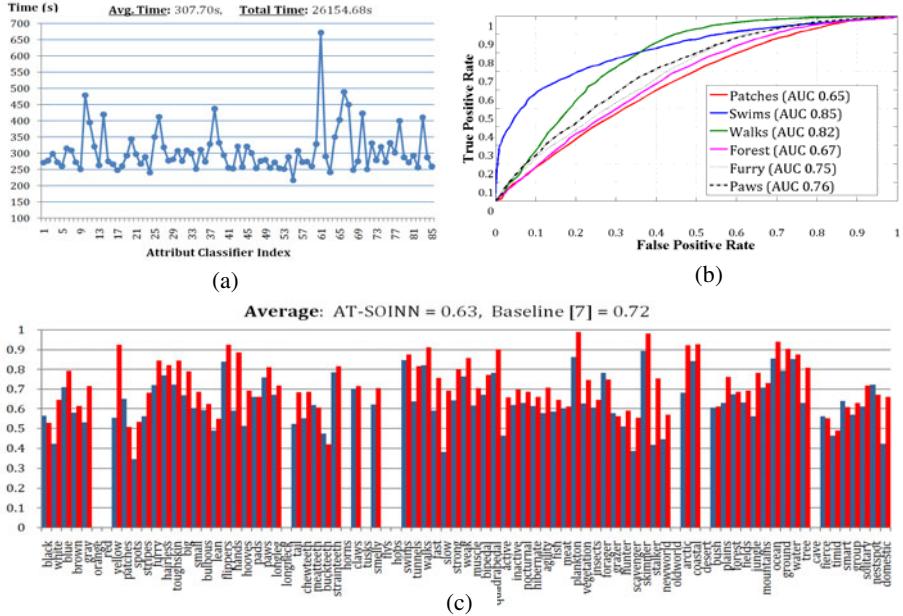


Fig. 3. The graph shows computation time (a) and typical area under ROC curve (AUC) (b). (c) Comparison of quality of individual attribute classifier as measured by AUC between AT-SOINN and [7]. Attribute with zeros entries have constant values for all test classes, so their AUC is undefined. Red and blue bar are of [7] and AT-SOINN, respectively. We confirm again that this result is obtained by using only 1 features out of 6 features used in [7].

Acknowledgement. This study was supported by the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

1. Bart, E., Ullman, S.: Learning Novel Classes from a Single Example by Feature Replacement. In: Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition, CVPR (2005)
2. Li, F.F., Fergus, R., Perona, P.: One-shot Learning of Object Categories. IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) 28(4), 594–611 (2006)
3. Stark, M., Goesele, M., Schiele, B.: A Shape-based Object Class Model for Knowledge Transfer. In: Proc. IEEE Int'l Conf. Computer Vision, ICCV (2009)
4. Shen, F., Hasegawa, O.: An Incremental Network for On-line Unsupervised Classification and Topology Learning. Neural Networks 19(1), 90–106 (2006)
5. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing Objects by their Attributes. In: Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition, CVPR (2009)
6. Wang, G., Forsyth, D.: Joint Learning of Visual Attributes, Object Classes and Visual Saliency. In: Proc. IEEE Int'l Conf. Computer Vision, ICCV (2009)
7. Lampert, C., et al.: Learning to Detect Unseen Object Classes by Between-Class Attribute Transfer. In: Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition (2009)
8. Farhadi, A., Endres, I., Hoiem, D.: Attribute-Centric Recognition for Cross-category Generalization. In: Proc. IEEE Int'l Conf. Computer Vision, ICCV (2010)

Author Index

- Ababneh, Jehad II-519
Abdel Hady, Mohamed Farouk II-70
Abe, Shige II-1
Adamopoulos, Adam I-241
Adams, Rod II-448
Adomi, Masahir II-94
Affonso, Carlos III-441
Aguilera Venegas, Gabriel III-471
Aiolfi, Fabi II-49
Aler, Ricardo I-422
Alexandridis, Georgios III-198
Alippi, Cesare II-458
Alonso-Betanzos, Amparo I-60, III-11
Alonso, Jesús B. I-565
Alonso, Lucian II-479
Alonso, Serafín II-392
Ampazis, Nicholas III-429
Anastassopoulos, George C. I-241, I-500
Annuth, Hendrik III-228
Araújo, Aluizio F.R. II-296, II-420
Arce, Jesús II-479
Arie, Hiroaki II-256
Arik, Sabri I-575
Arslan, Emel I-575
Asai, Yoshiyuki I-145
Athanasiadis, Ioannis N. III-451
Augilius, Eimontas I-46
- Baccouche, Moez II-154
Balaraman, Ravindran II-210
Bantouna, Aimilia II-382
Barbero, Álvar II-30
Barnard, Philip III-504
Barron-Zambrano, Jose Hug II-276
Barth, Erhardt III-52
Baskurt, Atilla II-154
Beard, Daniel W. III-451
Behnke, Sven III-82, III-92
Ben Abdallah, Ahmed Chamshedine II-486
Benabdeslem, Khalid III-387
Benyettou, Abdelkader I-117
Bergmann, Urs II-414, III-72
Bhamber, Ranjeet II-448
- Blachnik, Marcin III-257
Bodén, Mikael I-317
Bogdan, Martin I-184, I-211, II-160
Bohn, Christian-A. III-228
Bolis, Dimitris I-470
Bolón-Canedo, Verónica III-11
Boracchi, Giacom II-458
Borges, Rafael V. II-104
Boscolo, Sonia II-448
Bouboulis, Pantelis II-11
Bouchain, David I-311
Bowman, Howard III-504
Bozas, Konstantinos II-362
Brabec, Karel III-483
Brænne, Ingrid I-337
Braun, Jochen III-510
Broda, Krysia I-301
Brohan, Kevin II-180
- Camacho, Andres I-199
Canuto, Anne I-411
Canuto, Anne M. III-137
Castellanos Sánchez, Claudio I-188
Cataldo, Edson III-149
Cernega, Daniela Cristina II-286, II-307
Chakravarthy, Srinivasa I-282, II-210, II-216
Chantas, Giannis I-480
Chen, Ning III-277
Cheung, Yiu-ming I-384, III-435
Cierniak, Robert II-138
Contassot-Vivier, Sylvain I-261
Cook, Matthew I-164, III-267
Cuadrado, Abel A. II-343
Cutsuridis, Vassilis I-107, I-230
Cuzzola, Maria I-217
- d'Avila Garcez, Artur II-104
DalleMole, Vilson L. II-296
Da San Martino, Giovanni II-49
Davey, Neil II-448
de Araújo, Daniel III-397
de Carvalho, Luís Alfredo V. III-516
Deigmöller, Joerg II-124
De Lathauwer, Lieven II-59

- Demestichas, Panagiotis II-382
 Díaz, Ignaci II-343, II-392
 Diez, Alberto B. II-343
 Dimitriadis, Stavros I. II-362
 Di Noi, Lucia II-372
 Dittes, Benjamin III-247
 Domínguez, Manuel II-343, II-392
 Donangelo, Raul III-516
 Dorr, Michael III-52
 Dorronsoro, José R. II-30, II-82
 Draganova, Chrisina II-402
 Duan, Lijuan I-174
 Duarte, João III-277
 Duch, Włodzisław III-257
 Duch, Włodzisław II-40
 Dudek, Piotr II-180
 Dúračková, Daniela III-114
 Dušek, Ladislav III-483
 Dutoit, Pierre I-135
- Eggert, Julian II-124
 Elghazel, Haytham III-387
 El-Laithy, Karim I-184, II-160
 Emmerich, Christian II-148
 Endo, Tetsur II-256
 Engel, Paulo Martins II-170
 Estévez, Pablo A. III-178
 Etminani, Kobra I-101
- Fairhurst, Michael I-411
 Fairley, Jacqueline I-436
 Fedele, Roberta I-217
 Fernandez-Lorenzo, Santiago I-415
 Fernández, Mónica II-479
 Fernández-Navarro, Francisco I-327
 Ferrer, Miguel A. I-565
 Fiasché, Maurizio I-217
 Filipović, Jovana I-569
 Fischer, Asja III-208
 Fontenla-Romero, Óscar III-11
 Fontenla-Romero, Oscar I-60, I-415
 Forero Mendoza, Leonardo Alfredo III-149
 Fox, Charles I-388
 Franzius, Mathias III-298
 Freire, Luis C. I-22
 Frigui, Hichem II-486
 Fritsch, Jannik III-247
 Fu, Yu I-174
 Fuertes, Juan J. II-343, II-392
- Gader, Paul II-486
 Gallardo-Caballero, Ramón III-106
 Galván, Inés María I-422
 Garcez, Artur d'Avila I-301
 Garcia, Christophe II-154
 Galán García, José Luis III-471
 García, Sandra I-422
 García-Manso, Antonio III-106
 García-Orellana, Carlos J. III-106
 Gaussier, Philippe II-317, III-345
 Georgiou, Harris V. I-251
 Georgiou, Olga I-442
 Georgoulas, George I-436
 Gepperth, Alexander III-247
 Girau, Bernard II-276
 Gnecco, Giorgio III-358
 Godinho, Fernando M. I-22
 Gómez, Vicente III-352
 González Bandala, Daniel Alejandro I-188
 González-Velasco, Horacio M. III-106
 Gori, Marco III-315
 Gouveia, Ana R. I-22
 Grabner, Helmut I-551
 Grahl, Miranda II-492
 Grandinetti, Lucio III-327
 Gravalos, Ioannis II-410
 Grim, Jiří III-31
 Grimaldi, Domenico I-521
 Gross, Horst-Michael I-362, II-190, II-222
 Grossberg, Stephen III-495
 Grüning, André I-224
 Grüttner, Mandy II-114
 Gschwind, Régine I-261
 Guan, Xudong I-193
 Guijarro-Berdíñas, Bertha I-415
 Guillaume, Alain I-272
 Günzel, Dorothee I-211
 Gurney, Kevin II-180
 Gutierrez, German I-50
 Gutiérrez, Pedro A. I-327
 Güttler, Frank I-184
- Hagenbuchner, Markus II-372, III-237
 Hagiwara, Masafumi III-102, III-286, III-368
 Hájek, Petr I-1
 Häming, Klaus II-200

- Hara, Kazuyuki III-339
 Haralambous, Haris I-32
 Hartmann, André II-222
 Hasegawa, Hiroaki III-102
 Hasegawa, Osamu III-521, III-528,
 III-535, III-541, III-551, III-563
 Hasselmo, Michael I-230
 Hasson, Cyril II-317
 Hattori, Yuya I-401
 Hauser, Florian I-311
 Heinen, Milton Robert II-170
 Henriet, Julien I-261
 Hervás-Martínez, César I-327
 Hirano, Akihiro I-205
 Hirel, Julien III-345
 Hishida, Takanobu I-490
 Hoffmann, Jörn I-184
 Holoubek, Ivan III-483
 Honkela, Tim II-432
 Honkela, Timo I-351, I-368
 Hora, Jan III-31
 Horita, Hiroki I-205
 Hosino, Tikara I-77
 Huang, Jau-Chi III-282
 Hunt, Stephen II-448
 Husbands, Phil II-245
 Hřebíček, Jiří III-483
 Hyvärinen, Aapo I-67
- Iacopino, Pasquale I-217
 Iacucci, Ernesto I-267
 Ibarra, Manuel II-479
 Igel, Christian III-208
 Igual, Jorge I-199, II-519
 Ikeda, Kazushi III-321
 Iliadis, Lazaros II-21
 Iliadis, Lazaros S. I-241, I-500
 Imoto, Seiya I-67
 Inoue, Hirotaka III-557
 Irigoyen, Eloy III-352
 Ishiguro, Hiroshi III-407
 Ishii, Shin I-155, II-94
 Iwata, Akira I-490
- Jaeger, Siegfried III-464
 Jaziri, Rakia III-387
 Jeong, Sungmoon II-256
 Jia, Hong III-435
 Jirsa, Viktor II-353
- Joublin, Frank II-492
 Jug, Florian I-164
- Kaltenhaeuser, Robert I-362
 Kamimura, Ryotar II-333
 Karandashev, Yakov II-525, III-41
 Karapetsas, Eleftherios II-327
 Karathanasopoulos, Andreas S. I-428
 Karatzas, Kostas III-457, III-464
 Kateris, Dimitrios II-410
 Kawewong, Aram III-528, III-563
 Kayani, Umer I-356
 Kazemian, Hassan B. III-378
 Keck, Christian III-21
 Kivimäki, Ilkka I-368
 Klement, Sascha II-88
 Klós, Marzena I-42
 Kobayashi, Yasuhiko I-291, I-401
 Kobialka, Hans-Ulrich I-356
 Kobos, Mateusz III-124
 Kogure, Masumi I-127
 Kolodziejksi, Christoph I-374
 Komatsu, Yusuke III-309
 Komendantskaya, Ekaterina I-301
 Koprinkova-Hristova, Petia II-438
 Kortkamp, Marco III-188
 Kosmopoulos, Dimitrios I-551
 Koutitas, George III-477
 Krajmer, Mário III-114
 Krautz, Christoph I-164
 Krishnan, Ravi I-282
 Kryzhanovsky, Boris II-525, III-41
 Kubásek, Miroslav III-483
 Kučerová, Anna I-347
 Kugler, Mauricio I-490
 Kukolj, Dragan D. I-569
 Kummert, Franz II-492
 Kurek, Jerzy E. II-266
 Kůrková, Věra III-358
 Kurokawa, Hiroaki III-110
 Kuroyanagi, Susumu I-490
 Kyriakidis, Ioannis III-457
- Laaksonen, Jorma I-531
 Labusch, Kai I-337
 Lagus, Krista I-351, I-368
 Lalov, Boyan III-303
 Lamb, Luis C. II-104
 Lamonaca, Francesco I-521
 Lanaridis, Aris II-531

- Larrea, Mikel III-352
 Laskaris, Nikolaos A. II-362
 Laskowski, Lukasz III-294
 Laurent, Rémy I-261
 Lecluse, Michel I-117
 Lee, Minh II-256
 Lee, Xiaodong III-118
 Leloudas, Panagiotis M. I-428
 Letosa, Jorge Ramón II-432
 Lezoray, Olivier I-117
 Li, Meng I-384
 Likas, Aristidis I-87
 Likothanassis, Spiridon D. I-428
 Lindh-Knuutila, Tiina I-351
 Liou, Cheng-Yuan III-282
 Litinskii, Leonid III-41
 Liu, Qingshan II-498
 Llinares, Raul I-199, II-519
 López, Jorge II-82
 Loukas, Athanasios II-21
 Loutridis, Spyridon II-410
 Løvlid, Rikke Amilde III-143
 Lücke, Jörg III-21
 Macías-Macías, Miguel III-106
 Madany Mamlouk, Amir I-337
 Maggini, Marco III-218
 Makibuchi, Naoya III-521, III-551
 Makovicka, Libor I-261
 Mamalet, Franck II-154
 Manaithunai, Maya II-210
 Mańdziuk, Jacek III-124
 Manoonpong, Poramate I-374
 Mantzaris, Dimitrios I-241
 Mareš, Tomáš I-347
 Margaritis, Konstantinos G. I-395
 Maris, Fotis II-21
 Markos, Angelos I-395
 Maronidis, Anastasios I-460, I-470
 Martinetz, Thomas II-88, III-52
 Martínez-Rego, David I-60
 Martín-Merino, Manuel III-62
 Martins, Allan III-397
 Maszczyk, Tomasz II-40
 Matsumoto, Yoshio III-407
 Matsuzaki, Shuichi I-127, I-511
 Matuzevičius, Dalius I-541
 Mavroforakis, Michael E. I-251
 Mayer, Rudolf II-426
 Mažgut, Jakub I-317
 Meftah, Boudjelal I-117
 Melacci, Stefano III-315
 Melo, Jorge III-397
 Mérida Casermeiro, Enrique III-471
 Mesin, Luca III-489
 Metin, Selin II-228
 Metta, Giorgi II-234
 Miao, Jun I-174
 Miller, Bartosz I-97
 Miro-Borras, Juli II-519
 Miro-Borras, Julio I-199
 Mitianoudis, Nikolaos I-450
 Mitrakis, Nikolaos III-464
 Miyoshi, Seiji III-339
 Mohan, Vishwanathan II-234
 Moioli, Renan C. II-245
 Morabito, Francesco C. I-217
 Morán, Antoni II-392
 Morasso, Pietr II-234
 Moreau, Yves I-267
 Moshou, Dimitrios II-410
 Mouraud, Anthony I-272
 Možaryn, Jakub II-266
 Müller, Andreas III-92
 Muratore, Donatella III-237
 Mycka, Lukasz I-54
 Naghibzadeh, Mahmoud I-101
 Nakagawa, Masahiro III-411
 Nakagawa, Masanori II-142
 Nakamura, Yuichi III-411
 Nakamura, Yutaka III-407
 Nakano, Ryohei III-290
 Nakayama, Kenji I-205
 Naoki, Honda I-155
 Navakauskas, Dalius I-541
 Neto, Abner Rodrigues III-423
 Neto, Adrião Dória III-397
 Neto, Antonino Feitosa I-411
 Neto, Guenther Schwedersky III-423
 Neves, João III-277
 Nieminen, Ilari T. I-368
 Nishida, Toyoaki II-468, III-541
 Okada, Shog II-468
 Okada, Shogo III-541
 Olej, Vladimír I-1
 Ono, Katsuya III-339
 Ordoñez, Víctor II-479
 Orione, Fiammetta III-489

- Ortiz, Michaël Garcia III-247
 Osendorfer, Christian III-168
 Osowski, Stanislaw I-54
 Othman, Ahmed A. I-561
 Öztürk, Pinar III-143
- Paliy, Ihor I-521
 Palmer-Brown, Dominic II-402
 Palm, Guenther II-438
 Palm, Günther I-311, II-70
 Papadopoulos, Harris I-32
 Papadourakis, George III-457
 Papini, Tiziano III-218
 Parviainen, Elina III-1
 Pasemann, Frank I-374
 Pasero, Eros III-489
 Pastukhov, Alexander III-510
 Patan, Krzysztof II-134
 Patan, Maciej II-134
 Paugam-Moisy, Hélène I-272
 Peralta, Juan I-50
 Perdikis, Dionysios II-353
 Pérez-Oria, Juan II-479
 Pérez-Sánchez, Beatriz I-60
 Perrig, Stephen I-135
 Peters, Gabriele II-200
 Petrakis, Stelios III-417
 Picton, Phil III-378
 Pimenidis, Elias II-21
 Pintro, Fernando I-411
 Pitas, Ioannis I-460, I-470, III-333
 Pokrić, Maja I-569
 Popa, Cosmin I-12
 Porto-Díaz, Iago III-11
 Prada, Miguel A. II-343, II-392
 Prescott, Tony I-388
 Principe, Jose C. III-178
- Qing, Laiyun I-174
 Quoy, Mathias III-345
- Raschman, Emil III-114
 Ratnadurai, Shivakesavan I-282
 Rauber, Andreas II-426
 Razavi, Amir Reza I-101
 Reguera, Perfect II-392
 Reinhart, René Felix II-148
 Remy, Guillaume F. III-378
 Ribeiro, Bernardete III-277
 Riquelme, José C. I-327
- Rizzoli, Andrea-Emilio III-451
 Rodríguez Cielos, Pedro III-471
 Rodríguez, Cristina II-479
 Rodriguez, Francisco B. II-506
 Roisenberg, Mauro III-423
 Rolfe, Jason T. III-267
 Roveri, Manuel II-458
 Rozado, David II-506
 Rührmair, Ulrich III-168
 Ruiz, Roberto I-327
 Rye, David I-436
- Sachenko, Anatoly I-521
 Sagara, Tsukasa III-286
 Saito, Kazumi III-290
 Sakashita, Tetsuya I-291
 Salomon, Michel I-261
 Sánchez-Marcano, Noelia I-415
 Sánchez Orellana, Pedro Luis I-188
 Sanchis, Araceli I-50
 Sanguineti, Marcello III-358
 Sano, Toshifumi I-511
 Santana Jr., Orivaldo V. II-420
 Santana, Laura E.A. I-411, III-137
 Santos, Araken M. III-137
 Sassi, Renato Jose III-441
 Sauget, Marc I-261
 Sawalhi, Nader II-410
 Scarselli, Franc II-372
 Scarselli, Franco III-237
 Schaffernicht, Erik I-362, II-190, II-222
 Schaul, Tom II-114
 Schels, Martin II-70
 Scherer, Dominik III-82, III-92
 Schmidhuber, Jürgen II-114, III-168
 Schmid, Thomas I-211
 Schulz, Hannes III-82
 Schwenker, Friedhelm II-70
 Sehnke, Frank II-114, III-168
 Şengör, Neslihan Serap II-228
 Serackis, Artūras I-541
 Serbencu, Adriana II-286
 Serbencu, Adrian Emanoil II-286
 Shafarenko, Alex II-448
 Shaposhnyk, Vladyslav I-135
 Shen, Fura II-76
 Shen, Furao III-521, III-535, III-551
 Shikauchi, Yumi II-94
 Shimamura, Teppei I-67
 Shimizu, Shohei I-67, III-309

- Shimodaira, Hidetoshi III-309
 Signoretto, Marc II-59
 Silva, Marco III-149
 Siolas, Georgios III-198
 Sirola, Miki I-46
 Siwek, Krzysztof I-54
 Slater, Brendan II-448
 Sogawa, Yasuhiro I-67
 Soh, Zu I-401
 Solea, Razvan II-307
 Sölder, Jan III-168
 Soussen, Charles I-261
 Spartali, Iliana II-21
 Sperduti, Alessandr II-49
 Stafylopatis, Andreas II-531, III-198
 Stamatis, Demosthenes II-327
 Stavropoulos, Thanos G. III-477
 Steege, Frank-Florian II-222
 Steger, Angelika I-164
 Steil, Jochen Jakob II-148
 Stephanakis, Ioannis M. I-500
 Stylios, Chrysostomos I-436
 Subramanian, Deepak I-282
 Sugano, Shigeki II-256
 Sukumar, Deepika II-216
 Su, Li III-504
 Sun, Yi II-448
 Suykens, Johan A.K. II-59
 Suzuki, Michiyo I-291, I-401
 Swiderski, Bartosz I-54
- Tachos, Stavros II-21
 Takemura, Noriko III-407
 Takenouchi, Takashi III-321
 Takeuchi, Johane II-204
 Talonen, Jaakko I-46
 Tanahashi, Yusuke III-290
 Tangruamsub, Sirinart III-528
 Tani, Jun II-256
 Taormina, Riccardo III-489
 Taylor, John G. III-496
 Tefas, Anastasios I-460, I-470,
 III-333, III-417
 Temerinac, Miodrag I-569
 Terai, Asuka II-142
 Theodoridis, Sergios I-251, II-11
 Theofilatos, Konstantinos A. I-428
 Tiño, Peter I-317
 Tizhoosh, Hamid R. I-561
 Torreão, José R.A. I-305
- Torres-Huitzil, Cesar II-276
 Travieso, Carlos M. I-565
 Tsagkaris, Kostas II-382
 Tsalkidis, Aggelos I-241
 Tsapanos, Nikolaos III-333
 Tsapatsoulis, Nicolas I-442
 Tscherepanow, Marko III-157
 Tsinaslanidis, Prodromos III-130
 Tsoliaridou, Ageliki III-477
 Tsoi, Ah Chung II-372, III-237
 Tsuboyama, Manabu III-528
 Tsujino, Hiroshi II-204
 Tsuji, Toshio I-291, I-401
 Turchenko, Volodymyr I-521, III-327
 Turitsyn, Sergei K. II-448
 Tzelepi, Areti II-362
 Tzikas, Dimitris I-87
 Tzikopoulos, Stylianos D. I-251
- Urbánek, Jaroslav III-483
- Van Gool, Luc I-551
 Vargas, Patricia A. II-245
 Varona, Pablo II-506
 Varvarigou, Theodora I-551
 Vassileiou, Apostolos II-21
 Väyrynen, Jaakko J. I-368
 Vellasco, Marley III-149
 Vera, Pablo A. III-178
 Verma, Saurabh Shekhar I-362
 Victer, Silvia M.C. I-305
 Vieira, Armando III-277
 Vig, Eleonora III-52
 Viitaniemi, Ville I-531
 Villa, Alessandro E.P. I-135, I-145
 Vlahavas, Ioannis III-477
 Vollmer, Christian II-190
 Vonau, Victoria III-510
 von der Malsburg, Christoph II-414,
 III-72
 Voulodimos, Athanasios I-551
 Vozalis, Manolis G. I-395
 Vrakas, Dimitris III-477
- Wachsmuth, Sven III-188
 Wada, Yasuhiro I-127, I-511
 Wang, Jun II-498
 Wang, Xin III-118
 Washio, Takashi I-67
 Waszczyzyn, Zenon I-42

- Wedemann, Roseli S. III-516
Wei, Hui I-193
Wersing, Heiko III-298
Willert, Volker II-124
Wolf, Christian II-154
Woodman, Marmaduke II-353
Wörgötter, Florentin I-374
Wu, Jun III-118

Yabuwaki, Ryousuke II-1
Yamane, Hiroaki III-368
Yamao, Masataka I-155
Yamasaki, Kazuhiro III-521
Yan, Baoping III-118
Yan, Hong I-317

Yonekawa, Masato III-110
Yu, Hui II-76
Yusoff, Nooraini I-224

Záluský, Roman III-114
Zapranis, Achilleas III-130
Zarzoso, Vicente II-519
Zenzeri, Jacop II-234
Zhao, Jinxi II-76
Zheng, Jun II-76
Zhu, Junmei II-414
Zlokolica, Vladimir M. I-569
Zou, Baixian I-174
Zuo, Qingsong I-193